José Rafael Ferreira Lucas Simão

Licenciado em Engenharia Electrotécnica e de Computadores

Aplicação educativa para a promoção da linguagem em crianças com perturbação do desenvolvimento

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Yves Rybarczyk, Professor Doutor, FCT

Co-orientador: Tiago Cardoso, Professor Doutor, FCT

Júri:

Presidente: [Nome do presidente do júri]

Arguentes: [Nome do arguente 1]

[Nome do arguente 2]

Vogais: [Nome do vogal 1]

[Nome do vogal 2]

[Nome do vogal 3]

[Nome do vogal 4]

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2016

Aplicação educativa para a promoção da linguagem em crianças com perturbação do desenvolvimento

Agradecimentos

A realização desta dissertação é o ponto que marca a finalização do meu percurso académico ao qual dediquei anos da minha vida. Durante a sua duração encontrei diversas pessoas que me auxiliaram e me orientaram culminando no resultado atual, sem elas esta dissertação não seria possível.

Gostaria de agradecer primeiramente ao meu avô Artur Lucas Simão, que desde criança me demonstrou que as grandes conquistas são alcançadas pela simplicidade, persistência e dedicação das pessoas, obrigado por todo o apoio incondicional que me forneceu ao longo de toda a minha vida. Ao meu pai, que nunca deixou de me apoiar e sempre arranjou maneiras de me motivar e de me guiar sem nunca me negar ou vacilar qualquer pedido. À minha mãe, pelo amor e motivação incondicional.

Ao José Franco, que demonstrou ser uma pessoa dedicada, um guia e um amigo que nunca desistiu de mim e que sempre me ajudou durante esta jornada. Ao meu afilhado Frederico Monteiro, que foi um colega excecional e me apoiou neste percurso. Ao meu amigo Pedro Fernandes, que me proporcionou uma nova maneira de pensar e que me ajudou a descobrir soluções, mesmo quando parecia não existir alguma. Ao Gonçalo Costa, que mostrou um grau elevado de companheirismo e uma maneira de ser admirável. À Doutora Luísa Cotrim e à Doutora Teresa Condeço, que depositaram todas as suas expectativas em mim, pelo apoio fornecido e pelo tempo disponibilizado para a realização desta dissertação. À Isabel Lima, que me proporcionou uma visão madura do mundo e bastante apoio em áreas fora do meu domínio. Aos meus amigos e colegas de curso que me acompanharam durante a minha vida académica, André Lourenço, António Abel, Bruno Letras, Bruno Nascimento, Bruno Gonçalves, Bruno Rodrigues, Daniel Nunes, David Cantarinha, Diogo Cardoso, Diogo Silva, Diogo Teixeira, Filipa Pinheiro, Filipe Carrasquinho, Filipe Correia Carrasquinho, Filipe Santiago, Filipe Viegas, Francisco Coito, Gonçalo Mestre, Iuri Rodrigues, João Alarcão, João Alexandre, João Santos, Joana Simão, José Marques, Nataniel João, Mariana Côrte-Real, Miguel Fernandes, Ricardo Bernardo, Ricardo Ganilha, Ricardo Mexia, Rui Guerra, Rui Mausinho, Rui Silva, Rúben Santinhos, Sérgio Coelho, Sérgio Saro, Sofia Ribeiro e Tânia Carvalho.

À Universidade Nova de Lisboa e à Faculdade de Ciências e Tecnologias por me terem fornecido todo o conhecimento e desenvolvimento profissional, disponibilizando as suas instalações e recursos para me formar e me preparar para o futuro. Ao Centro Diferenças e especialmente às crianças que o frequentam pelo tempo prestado, interesse, disponibilidade e dedicação para apoiar o desenvolvimento desta dissertação.

Para finalizar, gostaria de agradecer ao meu orientador, Professor Yves Rybarczyk, e ao co-orientador, Professor Tiago Cardoso, por me terem proporcionado a oportunidade de realizar esta dissertação, assim como, a motivação, dedicação e apoio fornecido ao longo do seu desenvolvimento.

Muito Obrigado!

Resumo

Atualmente, a quantidade de dispositivos e tecnologia existentes podem criar a diferença no auxílio ao ensino, especialmente de crianças com necessidades especiais. Contudo, apesar da empatia que a sociedade coloca no uso da tecnologia e a sua rápida expansão na área da educação, poucas são as iniciativas que se focam nas necessidades das crianças com dificuldades na aprendizagem. Embora existam diversas empresas com capacidade monetária para suportar e criar uma nova ferramenta que auxilie estes alunos, raramente o investimento é rentável o suficiente para proporcionar a continuação da criação destas ferramentas e tecnologias.

Com o objetivo de prevenir e preencher esta falha existente, esta dissertação propõe-se a dar uso às tecnologias existentes, aliando-se aos conhecimentos adquiridos na instituição Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa de modo a criar uma ferramenta que ajude os estudantes que lidam com diversas dificuldades de aprendizagem. Esta dissertação conta com a parceria do Centro Diferenças – Centro de Desenvolvimento Infantil, instituição esta, que tem como objetivo principal trabalhar e ajudar crianças com dificuldades no neuro-desenvolvimentos, nomeadamente com síndrome de *Down*. Esta dissertação dá uso à sua parceria para criar uma ferramenta que irá trabalhar áreas específicas relacionadas com a consciência fonológica, de maneira a facilitar a compreensão e a absorção de conhecimento por parte dos alunos, aproveitando as vantagens que um jogo fornece aos seus jogadores. Esta ferramenta irá ser um conjunto de mini-jogos que irão endereçar aspectos diversos no ensino a crianças com problemas no neuro-desenvolvimento.

Palavras-chave: síndrome de Down, Motor de jogo, jogo educacional, neuro-desenvolvimento, multiplataforma.

Abstract

Currently the number of devices and existing technology can make a difference in learning and education, especially with children with special needs. However, despite the empathy that society places on the use of technology and its rapid expansion in the area of education, there have been few initiatives that focus on the needs of children with learning difficulties. Although there are companies that may have the financial means to support and provide for the creation of new tools to assist in the learning and development of these children, rarely does the investment prove itself profitable enough to provide for their continued development.

In order to prevent and address this missing gap, this dissertation proposes the use of existing technologies and knowledge base within the institution *Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa* to create a tool to help children with different learning disabilities. This thesis included a partnership with the *Centro Diferenças* - Child Development Centre, an Institution which focuses on working and helping children facing neuro-development limitations, particularly those with Down syndrome. This dissertation utilizes this partnership to create a tool that will work with specific areas related to phonological awareness, in order to facilitate a student's overall level of understanding and knowledge absorption, whilst at the same time taking into account the benefits that a game may provide to its players. This tool will be formed by a set of mini-games that will target specific areas of the teaching in children with Down syndrome.

Keywords: Down syndrome, Game Engine, educational game, neuro-development, multiplatform.

Índice de Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Símbolos e Acrónimos

2D	Duas Dimensões
3D	Três Dimensões
API	Application Programming Interface
HTML	HyperText Markup Language
LAN	Local Area Network
JSON	Javascript Object Notation
PC	Personal Computer
UML	Unified Modeling Language
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
UI	User Interface
WebGL	Web Graphics Library
XML	Extensible Markup Language

1

Introdução

Atualmente a tecnologia está acessível a tudo e todos e existem diversos dispositivos espalhados pela casa, nos escritórios, nos cafés e locais de convívio, entre outros. Em Portugal, durante o ano de 2015 registrou-se que 70% das famílias portuguesas têm acesso à internet a partir de casa, entre as quais, 66% acedem à internet fora de casa e do local de trabalho recorrendo ao uso de dispositivos móveis (Instituto Nacional de Estatística, 2015). Tendo em conta a grande quantidade de dispositivos à disposição da maioria da população, é apenas lógico que se use a tecnologia para resolver diversas problemáticas que ocorrem no dia-a-dia.

Devido às grandes vantagens que os dispositivos móveis apresentam, tais como, portabilidade, velocidade, eficácia e acessibilidade, existe um grande interesse em desenvolver diversas funcionalidades e aplicações que permitam o acesso de conteúdo em diversas plataformas, computador, *smartphone*, *tablet*, entre outros.

Uma das áreas que tem sido bastante explorada e que está constantemente à procura de complementos para auxiliar a educação é o ensino. Neste momento, com o aumento da quantidade de utilizadores que possuem acesso à internet, existem universidades *online*, em que um aluno não necessita de frequentar fisicamente uma universidade para obter um grau académico. As aulas são dadas usando o auxílio de um dispositivo de captura de imagem, que gravam o professor e a aula é dada *online* em tempo real, sendo as dúvidas tiradas num fórum ou por videoconferência. Embora o método nomeado anteriormente não seja o mais usado, a sua popularidade tem vindo a crescer, aumentando-se assim o ensino *online*. No entanto, este tipo de ensino embora seja uma alternativa positiva, comparando com o ensino praticado anteriormente (aulas presenciais), apresenta vantagens, tais como, mobilidade e dinamismo, mas também apresentam algumas desvantagens, como uma maior dificuldade de cativar a audiência (no caso de o professor estar incapacitado de ver a audiência ou no caso da aula ser gravada para futuras visualizações) juntamente com um menor controlo sobre o conhecimento dos alunos durante a aula (perde-se a capacidade de verificar se o aluno percebeu ou se está atento durante a aula). Os jogos por outro lado fornecem diversas vantagens que conjugadas à educação transformam atividades e informação anteriormente difíceis de acumular, em material interessante devido à motivação inata que os jogos fornecem juntamente com a capacidade de simulação que permite o aluno errar e aprender com os erros, sem sentir qualquer preocupação quando erra, focando-se assim em tentar atingir um objetivo.

Durante anos, as empresas desenvolveram jogos educativos, com o objetivo de cativar os alunos a aprender enquanto jogam. Contudo, embora estes jogos estivessem construídos para a maioria dos alunos, existe uma pequena porção que não poderia usufruir dos jogos educativos genéricos visto possuírem algum tipo de problema no neurodesenvolvimento o que aumenta a dificuldade na aprendizagem. Só em Portugal, 1 em cada 800 crianças nasce com a síndrome de *Down*, estas crianças apresentam dificuldades de atenção e problemas diversos no neuro-desenvolvimento (Pais21, 2016). No entanto, as ferramentas eletrónicas existentes para auxiliar a aprendizagem de crianças que apresentam problemas no neurodesenvolvimento revelaram ser reduzidas e bastante limitadas. Existem diversas empresas que possuem capacidades para criar e

fornecer ferramentas, contudo, o público-alvo é reduzido em termos quantitativos, o que por vezes, não justifica o investimento.

Esta problemática motiva associações especializadas a criarem parcerias com institutos superiores, com o objetivo de colmatar a falha existente na quantidade de ferramentas disponíveis e numa tentativa de encontrar alternativas que melhorem a educação das crianças de modo a permitir que estas consigam integrar-se com sucesso na sociedade. Um exemplo deste tipo de iniciativa é a parceria entre a Universidade Nova de Lisboa – Faculdade de Ciências e Tecnologias e o Centro Diferenças, onde esta dissertação se insere, pretendendo com esta, criar uma aplicação que contenha um conjunto de atividades ou minijogos que irão ter como objetivo, auxiliar o ensino de pessoas com problemas no neurodesenvolvimento.

1.1 Objetivos

Numa fase inicial, foram delineados todos os objetivos a cumprir durante a elaboração desta dissertação, sendo estes:

- A criação de um sistema multi-plataforma que permita a gravação de dados dos utilizadores.
- Criar um videojogo que permita substituir diversas ferramentas, recorrendo ao uso de atividades ou minijogos para auxiliar no ensino para crianças com problemas no neurodesenvolvimento.
- Desenvolver um sistema que tenha a capacidade de ser executado em dispositivos móveis.
- Criar um sistema que permita verificar e avaliar o progresso do utilizador ao longo do tempo.

Embora o sistema seja uma prova de conceito bastante importante para esta dissertação, a componente que mostra mais pertinência é a de encontrar a resposta à questão:

Será que as vantagens presentes nas mecânicas dos jogos quando aplicadas a ferramentas de ensino para a promoção da linguagem vão realmente aumentar o desempenho da aprendizagem em pessoas com problemas no neurodesenvolvimento?

1.2 Público-Alvo

Esta dissertação é um dos produtos da parceria do Centro Diferenças com a Universidade Nova de Lisboa – Faculdade de Ciências e Tecnologias, com o objetivo de construir e fornecer ferramentas para apoiar pessoas com problemas no neurodesenvolvimento. Este projeto pretende promover o desenvolvimento da linguagem de modo a auxiliar a aprendizagem destas pessoas, assim como, facilitar a sua integração na sociedade através do conhecimento adquirido, durante o seu uso.

A síndrome de Down é um distúrbio genético descoberto pelo médico John Langdon Down em 1866 e é caracterizada pela existência de um cromossoma 21 adicional nas células do portador. As pessoas que sofrem desta síndrome apresentam diversos atrasos a nível mental e motor, sendo o atraso mental considerado uma das características mais constantes da Trissomia 21 (Moreira & Gusmão, 2002). Por norma, os termos "deficiência mental", "atraso mental" e "défice cognitivo" empregam-se com o objetivo de identificar um défice de rendimento e capacidade de integração num meio social. O método usado para avaliar o desenvolvimento cognitivo de uma criança ou de um jovem é sempre pelo seu "atraso" em comparação com crianças ou jovens que são classificados como "normais". Os portadores desta síndrome, com défice cognitivo, de um modo

geral apresentam dificuldades de abstração, generalização e ligação de ideias. Verifica-se que estes reagem melhor a imagens (pensamento concreto) do que a conceitos (pensamento abstrato), demorando mais tempo no pensamento concreto do que outras pessoas denominadas "normais". Em relação à área da linguagem, os portadores desta síndrome revelam dificuldades ao nível da fala, expressão, compreensão e síntese, juntamente com a organização de pensamento (Cunha & Santos, 2007). Contudo, existem trissómicos 21 que demonstraram grande capacidade de memorização devido às suas atividades, mas, genericamente, apresentam fraca capacidade de evocar informação previamente adquirida, o que se torna um obstáculo para a aprendizagem do vocabulário e suscita erros e falhas na construção gramatical da linguagem. É de notar que, o atraso no desenvolvimento da linguagem está diretamente relacionado com a lentidão que os trissómicos apresentam na sua organização de ideias, que está relacionada com o pensamento muito elementar que demonstra um défice da função simbólica, tornando-se assim difícil atingir o pensamento abstrato (Burgoyne, 2009).

Apesar do interesse da maioria dos doentes em aprender, a sua falta de concentração e a sua grande propensão para o cansaço e fadiga apresentam uma grande barreira no caminho da aprendizagem (Cunha & Santos, 2007). Com o objetivo de resolver a problemática anterior, foram desenvolvidas diversas ferramentas de modo a auxiliar a sua aprendizagem. Contudo, as ferramentas e consultas de neurodesenvolvimento existentes provaram ser escassas para a resolução do problema em mãos, segundo o Dr. Miguel Palha, "Após o nascimento da minha filha …vi-me confrontado, em matéria de apoios ao seu desenvolvimento, com dificuldades inesperadas: as consultas de neurodesenvolvimento, para além de escassas, não eram especializadas" (Diferenças, 2016). Embora o nascimento da sua filha tenha sido há um período de tempo considerável, a problemática mantêm-se. Uma das razões é a falta de motivação de empresas em fabricar ferramentas especializadas. A solução que vai ser apresentada pretende ir ao encontro deste problema aumentando as já existentes, sendo esta especializada para o apoio de consultas de neurodesenvolvimento.

1.3 Motivação

O desenvolvimento de uma aplicação que dê acesso a pessoas com problemas no neurodesenvolvimento um conjunto de ferramentas que as auxilie na aprendizagem e que permita o acesso em qualquer local é um desafio suficiente para provocar interesse neste tema. A possibilidade de poder lidar com profissionais ligados a outros ramos, com diferentes experiências e opiniões possibilitando assim uma experiência única ainda não experienciada anteriormente. Contudo, o argumento que acabou por dissipar quaisquer dúvidas que existissem na decisão foi o uso desta ferramenta que uma vez finalizada será usada pela instituição Centro Diferenças para o ensino das crianças com problemas no neurodesenvolvimento.

O conhecimento por si só deve ser acessível a todos, sendo esta uma oportunidade de quebrar uma barreira que limita o ensino destas crianças. Sendo o resultado, uma experiência diferente, interessante e gratificante.

2

Estado de Arte

Neste capítulo será apresentado o resultado da pesquisa feita sobre os jogos eletrónicos de *browser*. Irá ser dada mais importância a jogos de *browser* com componente educativa, visto que é o que se adequa mais ao tema em questão.

Irão ser identificados os web services que apresentam vantagens para serem implementados em jogos de browser. Também serão apresentadas as ferramentas mais usadas para o desenvolvimento deste tipo de jogos, juntamente com as respetivas vantagens e desvantagens.

2.1 E-Learning

O uso da tecnologia para facilitar a aprendizagem pode ser conhecido por *E-Learning*. A entrega do material de estudo, pode ser realizada de uma forma assíncrona, via servidores de Internet e *web browsers* (Horachek, 2014).

Atualmente todo o material de *E-Learning* está presente na internet, ao contrário do passado, em que todo o tipo de material de aprendizagem no computador teria de ser visualizado através de métodos como, por exemplo, o uso de um *CD-ROM*. Este oferece a possibilidade de se partilhar todo o material numa diversidade de formatos, tais como:

- Vídeos;
- *Slideshows*;
- Documentos Word;
- PDF.

Este modelo de aprendizagem também oferece aos professores a capacidade de fazerem *webinares* (aulas online em tempo real), onde os alunos podem comunicar com os professores, através do uso de um *chat* ou de uma mensagem nos fóruns. Os alunos têm assim, a possibilidade de aprender sem influenciar o seu estilo de vida. Desta forma, até a pessoa mais ocupada tem a capacidade de continuar a sua carreira com a vantagem de conseguir adquirir novas qualificações (Epignosis LLC, 2014).

Nos dias de hoje, o *E-Learning* é usado a nível mundial para se atingirem determinados objetivos educacionais em condições diferentes de educação (Kose & Arslan, 2015). Durante a pesquisa para esta dissertação, foram identificadas 3 gerações desta tecnologia:

- **Primeira geração** – Tinha como principal preocupação o uso passivo da internet.
- **Segunda geração** – Foi caracterizada por usar tecnologias mais avançadas como o *eAssessment* e os *Virtual Learning Environments*.
- **Terceira e última geração** – Focada no uso de ambientes de aprendizagem colaborativos, usa ferramentas como *wikis*, *blogs*, etc (Connolly & Stansfield, 2006).

Este modelo de ensino destaca-se dos seus competidores porque possuí diversas vantagens, tais como:

- **Não tem limites ou restrições** – Normalmente, tempo é um dos problemas para os estudantes e professores. No caso de aulas tradicionais (aulas presenciais), existe uma limitação de lugares e a quantidade de tempo que os alunos podem participar.
- **Mais divertido** – Como as aulas são desenhadas de uma maneira divertida e interativa, o fator de envolvimento do utilizador aumenta.
- **É Rentável** – Ao contrário dos livros e das aulas nos colégios e escolas, este tipo de material está a ser constantemente atualizado e não corre o risco de ficar obsoleto, sendo a sua implementação, bastante rentável a longo prazo.
- **É uma solução que se insere perfeitamente num problema do mundo atual** – Companhias e organizações estão constantemente a adotar novas tecnologias para aumentar a eficiência das operações do dia-a-dia, tornando assim o uso da internet uma necessidade (Epignosis LLC, 2014).

Como qualquer modelo, este também demonstra ter algumas desvantagens:

- **Risco de Isolamento** – A perda do ensino tradicional, implica também o desaparecimento da interação típica das aulas presenciais, fazendo com que o aluno interaja com outras pessoas com menos frequência, limitando assim a sua capacidade de trabalho de equipa e podendo até em caso extremo sofrer problemas de isolamento.
- **Problemas de Acessibilidade e Requisitos adicionais** – Um dos maiores problemas deste modelo, é a necessidade de aceder frequentemente a um computador com ligação à Internet, visto que nem todas as pessoas têm acesso a ambas as componentes. Embora a tendência seja a diminuição do custo destes componentes, ainda se pode verificar que nem todos têm a possibilidade financeira para suportar tais custos. Outro problema que se consegue identificar é que, caso existam problemas técnicos por parte do servidor, o aluno pode perder todo o trabalho e/ ou, estudo.
- **Lentidão na criação do material** – Um dos problemas que se encontra neste momento é a quantidade de pedidos para novos cursos e a incapacidade ou a falta de tempo para os fazer, visto que o material fornecido pelo E-Learning tem de preencher um conjunto de requisitos, com o qual os professores não estão habituados a lidar (O'Donoghue, Singh, & Green, 2004). Segundo Armstrong, L., "Um professor universitário que consegue manter a atenção de centenas de alunos numa aula, pode não ser capaz de replicar o mesmo resultado, quando o tenta fazer recorrendo ao uso da tecnologia" (Armstrong, 2000).

2.2 Jogos na Educação

2.2.1 Jogos Eletrónicos *Online*

Antes de definir o conceito de jogo eletrónico *online*, será mais importante responder a uma questão mais básica.

O que é um jogo?

Após alguma pesquisa na área, foi concluído que cada *game designer* tem uma definição de jogo diferente, sempre relativa à sua própria interpretação. Segundo Chris Crawford, "jogos são um subconjunto (*subset*) do entretenimento limitado a conflitos entre os quais os jogadores trabalham para alcançar os seus objetivos" (Crawford, 1984).

No entanto, Sid Meier, famoso programador e *designer* de bastantes jogos de estratégia, defende que "um jogo é uma série de decisões interessantes" (Rollings & Morris, 2004) . Finalmente segundo Salen & Zimmerman, "um jogo é um sistema em que os jogadores se envolvem num conflito artificial, definido por regras, no qual se pode obter um resultado quantificado" (Salen & Zimmerman, 2004).

Contudo, após uma pesquisa extensiva, a definição que foi selecionada para definir jogo nesta dissertação foi dita por Raph Koster, afirmando que:

> **"Jogos são puzzles para resolver, tal como tudo o que encontramos na vida. A única diferença entre jogos e a realidade, é que nos jogos os riscos são inferiores"** (Koster, 2006)**.**

Este também defende que os jogos, além de serem únicos e especiais, também são conjuntos concentrados de informação que exercitam o nosso cérebro culminando na seguinte citação:

> **"Jogos servem de ferramentas muito poderosas e são fundamentais para a aprendizagem"** (Koster, 2006)**.**

Existem diversos tipos distintos de jogos e de plataformas para os suportar, desde os tabuleiros ao computador, a diversidade é bastante grande. Como tal, dependendo das necessidades do utilizador, pode ou não existir uma alteração de plataforma. Tendo em conta que este capítulo tem como objetivo sumarizar todo o tipo de tecnologias que existem, decidiu-se que se deveria dar algum destaque ao conceito de videojogos e videojogos *online*.

Videojogos são qualquer forma de sistemas computacionais de entretenimento eletrónicos, quer textuais ou através do uso de imagens, usando qualquer plataforma eletrónica desde computadores até consolas que envolvem um ou mais jogadores num ambiente físico ou com comunicação em rede (Frasca, 2001). Uma vez definido o significado de videojogos em geral, decidiu-se mencionar um pouco sobre a sua história, de forma a demonstrar um conjunto de tecnologias intermédias que foram apresentadas, até ao resultado atual. O primeiro jogo de computador foi criado pelo Sr. William Higinbotham e era constituído apenas por um osciloscópio ligado a um computador que simulava um jogo de ténis de vista lateral. Este jogo, embora rudimentar, era bastante rápido e já simulava o movimento da bola influenciada pelo efeito da gravidade. No entanto, embora tenha sido um jogo inovador na altura, não serviu de inspiração para a indústria de videojogos. Steve Russel por outro lado foi considerado o pioneiro e impulsionador da indústria de videojogos com o jogo *Spacewar*, criado em 1961. O jogo tinha como objetivo sobreviver o máximo tempo possível enquanto o jogador era bombardeado com diversos elementos inimigos, desde naves espaciais a asteroides. Os movimentos estavam

limitados a uma rotação no sentido dos ponteiros do relógio e a respetiva rotação inversa, juntamente com a possibilidade de se mover para a frente (Rabin, 2010).

Uma vez lançado o jogo *Spacewar*, considerado o primeiro jogo interativo de computador, diversos jogos começaram a surgir, entre eles existem diversos títulos de clássicos como *Pong*, *Space Invaders* e *Galaxian Asteroids*, iniciando-se assim o crescimento da indústria de *Arcade*(figura 2.1). (The History of Videogames, Março, 1982).

Em 1977, verificou-se um salto tecnológico enorme na história dos videojogos com o lançamento da primeira consola, *ATARI VCS* (*Video Computer System*)(figura 2.3), conhecida hoje em dia por *ATARI 2600.* A meio do ano de 1982, iniciou-se o declínio das máquinas *Arcade* devido ao lançamento da nova consola da *Atari* e também do início de lançamento dos primeiros cartuchos que continham jogos conhecidos de *Arcade*, sendo um deles o *PAC-MAN* (lançado em Abril de 1982). Também nesse ano foi lançado o microcomputador ZX Spectrum(figura 2.2) de 8 bits (BBC, 2016) que íncluia jogos como o Football Manager(1982), Invaders(1982), Arcadia(1982), Frogger(1983), entre outros. A partir desse ano iniciou-se a subida da popularidade das consolas e, tal como tinha sido dito anteriormente, a queda das *Arcade*.

Figura 2.1 Arcade(Fonte: http://gizmodo.com)

Figura 2.2 ZX Spectrum(Fonte: www.c64vsspectrum.com).

Figura 2.3 ATARI VCS(Fonte: https://atariage.com/).

O ano de 1988 destacou-se porque foi atingido um novo pico de tecnologia nas consolas com o lançamento da primeira consola de 16 bits pela *Sega* e o primeiro *hybrid PC-Engine game console* pela NEC, servindo de pioneiro no desenvolvimento de jogos 2D, tais como: *Sonic* (1991)(Figura 2.4), *Super Mario World* (1990), *Battletoads & Double Dragon* (1993)(Figura 2.5), entre outros. Apenas em 1995 é que surgem as míticas consolas que abriram portas ao aparecimento de jogos com componente Tridimensional (3D). Nesse mesmo ano foram lançadas as consolas *Sega Saturn* e *Playstation*. A consola da Nintendo, *Nintendo 64* (Figura 2.6), foi revelada nesse ano, mas apenas lançada em 1996 (Kent, 2001).

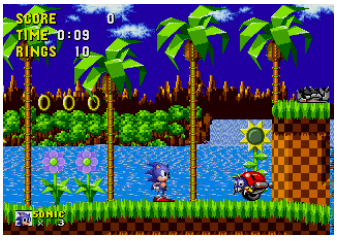


Figura 2.4 Sonic(1991)(Fonte: http://retrodrivegames.blogspot.pt/).

Figura 2.5 Battletoads & Double Dragon(Fonte: http://www.giantbomb.com/).

Figura 2.6 Nintendo 64(Fonte: www.old-computers.com).

Versões atuais destes sistemas, Playstation 4, Nintendo Wii U, entre outras, demonstram uma enorme evolução na componente gráfica, áudio e apresentam novas capacidades como, ligação à internet, grande armazenamento, venda de jogos sem necessidade da versão física do produto, entre outras (Sony, 2015) (Nintendo, 2015) (Microsoft, 2015). Conclui-se que a indústria de videojogos sofreu uma grande quantidade de evoluções, para uma indústria relativamente recente. Uma vez esclarecidos todos os conceitos básicos em relação ao jogo e aos videojogos em geral, pode-se então dar inicio à definição do que é um jogo eletrónico *online* ou um videojogo online.

Videojogos *online* estão associados a jogos *multiplayer* (multi-jogador) em que as máquinas dos jogadores estão ligadas por rede. Estes jogos não precisam de ser distribuídos através da internet, até os que são jogados recorrendo ao uso de uma LAN. No entanto, atualmente é uma prática bastante comum abandonar a compra da versão física do produto para se proceder à compra da versão virtual (Adams, 2010). Com esta pesquisa verificou-se que jogar este tipo de jogos, traz diversas vantagens. Perante os resultados recolhidos no final desta procura, decidiu-se mencionar as mais importantes:

- **Sociais** – Os jogos online oferecem a oportunidade de se interagir socialmente. A componente social é importante em diversos aspetos. Inicialmente, ajuda o jogador a construir relações sociais, normalmente para conquista de um certo objetivo ou para partilhar experiência de jogo. Se o objetivo desejado for atingido, normalmente os jogadores sentem-se inspirados e aumentam a sua ligação com a outra pessoa, o que pode ajudar a diminuir sentimentos de depressão.
- **Capacidade de Adaptação** – Este tipo de jogos faz com que os jogadores por vezes tenham de adotar "*Adaptive regulation strategies*" (estratégias adaptativas de regulação), ou seja, têm de aceitar que a sua estratégia não está correta e resolver o problema de outra forma. Se um jogo tiver a capacidade de mudar os desafios continuamente, como por exemplo, o jogo *Portal 2*[1], o jogador terá de estar constantemente a mudar a sua estratégia, visto que cada nível é diferente do anterior. Como as regras mudam rapidamente, os jogadores poderão ficar frustrados e até mostrar certos níveis de ansiedade, uma vez que são forçados a "desaprender" as estratégias usadas anteriormente e pensarem sobre a nova estratégia.
- **Regulador de Emoções** – Estudos correlacionais sugerem que estes jogos podem regular emoções (Olson, 2010). Os jogadores também relatam, após pensarem retrospetivamente, que as emoções positivas que sentiam quando estavam a jogar serviam de motivação para continuarem. Os jogos não fornecem apenas emoções positivas. Também enfurecem os jogadores, despertando raiva, ansiedade e tristeza. No entanto, se o jogador conseguir adaptar-se a essa situação e atingir o seu objetivo, sentir-se-á bem novamente, despertando assim sentimentos positivos.
- **Motivação** – Os *designers* de jogos são peritos em aproveitarem-se da capacidade de imersão de um jogo e arrastar pessoas para os jogar, fazendo e cumprindo objetivos insignificantes. Mantêm as pessoas a jogar, depois de falharem diversas vezes e ainda, conseguem que estas celebrem, quando cumprem um dos objetivos. Tudo isto porque os jogos têm a capacidade de fornecer feedback imediato, ou seja, quando um jogador atinge um objetivo, ele recebe automaticamente uma recompensa pelos esforços feitos. Desde conseguir destruir determinado castelo, a concluir um puzzle para prosseguir para a zona seguinte, entre outros (Granic, Lobel, & Engels, 2014).
- **Ausência de Problemas de disponibilidade** – Os jogadores conseguem sempre encontrar outras pessoas para jogar a qualquer hora do dia (Adams, 2010).

Uma vez apresentadas as vantagens, irão então ser nomeadas as desvantagens que foram selecionadas uma vez concluída a pesquisa para esta dissertação:

- **Problemas técnicos** – A qualquer momento pode falhar algo e sendo um jogo *online*, pode ser mais complicado de resolver do que se fosse um jogo local.
- **Latência** – Latência é um problema que é bastante importante visto que, não existem garantias de quanto tempo um pacote de dados vai demorar de um ponto a outro. Em

[1] Portal 2 é um jogo em que os jogadores têm de resolver puzzles recorrendo ao uso de portais. Com estes os jogadores podem manobrar objectos e movimentar-se pelo espaço de maneiras que seriam impossiveis de outra forma (Valve, 2016).

alguns jogos, ligações mais rápidas dão vantagens aos jogadores e fazem com que alcancem a vitória.

- **Perda e Recuperação de Pacotes** – É necessário a introdução de um mecanismo que detete a falta de um pacote de dados e peça o reenvio do mesmo, caso este seja perdido. Também se deve implementar um sistema que consiga resolver o problema dos pacotes chegarem fora de ordem, para evitar situações, como por exemplo, um jogador mover-se em frente no seu ecrã e depois ser teletransportado 5 metros atrás da posição onde ele pensava que estava.
- **Maus Comportamentos** – Lamentavelmente, visto que o jogador tem de falar com pessoas estranhas, neste caso em específico, pessoas estranhas e anónimas, isto pode permitir que um jogador use e abuse do seu anonimato para insultar, enganar ou até mesmo fazer batota sem se sentir muito ameaçado com as consequências. Para resolver este problema deve ser implementado um sistema que vigie e puna os jogadores que não cumprem as regras que foram impostas (Adams, 2010). No entanto, se analisarmos este sistema, podem-se identificar diversos problemas. Uma consequência da implementação deste sistema é que, se não for bem implementado, pode punir demasiado severamente jogadores que cometeram um erro pontual. Outro problema a ter em consideração é o facto de que, se o sistema começa a punir com demasiada frequência os jogadores, estes podem deixar de ter vontade de continuar a jogar.
- **Desperta emoções negativas nos jogadores** – Como foi dito anteriormente, estes jogos podem despertar emoções como a raiva, ansiedade e tristeza. Caso o jogador não seja capaz de lidar com a situação, estas emoções podem piorar, visto que os jogos fornecem recompensas a quem consegue superar os objetivos, mas, quem não consegue, não recebe tantas recompensas (Granic, Lobel, & Engels, 2014).

2.2.2 *Gamification*

Gamification é um conceito relativamente novo que foi introduzido por Nick Pelling em 2002 (Marczewski, 2013). Segundo Karl Kapp, este baseia-se no uso de elementos, mecânicas e pensamento de jogos num contexto sem relação aos mesmos, de maneira a tornar atividades diárias (aprendizagem, exercício físico, realidade aumentada) mais imersivas e apelativas (Enders, 2013).

No entanto, o conceito que foi selecionado para definir este conceito foi dito por Brian Burke que a define como "o uso de mecânicas e *design* para proporcionar uma experiencia motivante e imersiva com o propósito das pessoas concluírem certas metas ou objetivos " (Burke, 2014).

De modo a facilitar a visualização e a respetiva compreensão, decidiu demonstrar-se as diferenças entre Jogos e *Gamification* recorrendo à tabela 2.1.

Tabela 2.1 Diferenças entre Jogos e Gamification (fonte da tabela: http://www.gamification.org/education, 5/11/2015)

Jogos	***Gamification***
Têm um conjunto de regras e objetivos.	Pode conter uma coleção de tarefas que são pontuadas ou outro tipo de recompensa.
Existe a possibilidade de perder.	Perder é opcional, dependendo do tipo de aplicação. Pode não ser possível perder, isto porque normalmente o intuito é motivar as pessoas a cumprir uma tarefa ou um objetivo.
Normalmente apenas estar a jogar já é uma recompensa.	Ser recompensado por estar apenas a jogar é completamente opcional.
Requer um custo elevado para se elaborar.	Normalmente é mais fácil de elaborar e mais barato.
O conteúdo normalmente é feito e modificado para ser introduzido numa história e determinadas cenas do jogo.	Normalmente o conteúdo é acrescentado sem elaborar muitas modificações ao que já existia anteriormente.

Uma das grandes vantagens das aplicações que foram implementadas usando o conceito de *Gamification* é o uso equilibrado de motivações intrínsecas e extrínsecas de maneira a aumentar a motivação e o interesse. As motivações extrínsecas apoiam-se em recompensas externas, como por exemplo: pontos, bónus dentro de jogo, medalhas e conquista de títulos ou realizações (*Achievements*) (Marczewski, 2013). Relativamente às motivações intrínsecas, Ryan Richard e Deci Edward afirmam que estas referem-se à execução de algo, devido ao interesse ou ao agrado de um indivíduo. A pessoa é movida a fazer algo devido à diversão ou à satisfação e não devido a pressões ou recompensas. Alguns exemplos de motivações intrínsecas são: altruísmo, motivação, cooperação, amor ou agressão (Ryan & Deci, 2000).

Na figura 2.1, os círculos verdes significam o desejo primário que as mecânicas do jogo provocam enquanto os círculos azuis significam outros desejos que são afetados pela implementação dessas mecânicas. Em suma, motivações intrínsecas podem efetivamente ser geradas através de mecânicas de jogo, que implementam, e ou, representam motivações extrínsecas, como se pode verificar na figura 2.1.

2.2.3 *Serious Games*

O conceito ***Serious Games*** já existe há bastante tempo, no entanto apenas recentemente é que foi usado para o mundo digital, mais propriamente implementado em jogo eletrónicos. Clark Abt propôs em 1970 uma definição que ainda hoje é usada para descrever *Serious Games*. Este defende que "*jogos podem ser jogados de uma forma séria ou casual. No entanto levanta-se uma preocupação com os jogos sérios no sentido em que estes jogos estão explicitamente e cuidadosamente pensados com o objetivo educativo e não são desenhados primariamente para diversão. No entanto, isto não significa que os jogos sérios não são, ou não devem ser divertidos*" (Michael & Chen, 2006).

A definição que foi selecionada por estar mais sumarizada e ser bastante mais simples de entender, é a definição dada por David Michael e Sande Chen em que estes dizem, "Serious games são jogos que não tem entretenimento, gozo ou diversão como principal objetivo. Isto não significa que os mesmos não entretenham, dêem gozo de jogar ou divertidos. Apenas existe outro propósito, uma segunda intenção num sentido mais realista da questão". Contudo, deve ser reforçada a ideia de que, embora o nome *Serious Games* represente o propósito do jogo, a razão da sua criação, o conteúdo de um jogo deste género, pode não ter conteúdo sério. Em suma, o nome indica o propósito do jogo mas não dita ou rege o seu conteúdo (Michael & Chen, 2006).

Este tipo de jogos possuí um grande problema e/ou desafio que é conseguir que o jogador aprenda o conteúdo apresentado durante o jogo. Porém, caso este não seja apelativo, a pessoa perde a vontade de continuar a jogar, provocando lentamente o decaimento do interesse, atenção e vontade para aprender o conteúdo. Assim sendo, os *designers* e *developers* têm de ter a capacidade de nivelar o fator divertimento com a componente de ensino, de maneira a manter o jogador motivado e atento durante o processo de ensino.

Todavia, nem tudo são desvantagens. Os jogos eletrónicos possuem uma grande vantagem perante outros métodos: a capacidade de imersão. Esta possibilita uma ligação emotiva do jogador com uma personagem ou um mundo virtual. Esta ligação permite, com bastante facilidade, a simulação de problemas e estratégias presentes na vida real num ambiente completamente controlado onde o utilizador tem menos "medo" de cometer erros e sem temer qualquer tipo julgamento, devido ao conforto do ambiente controlado (Michael & Chen, 2006). Em suma, o objetivo principal dos jogos sérios é ensinar algo e, se possível, ser divertido durante o processo.

2.2.4 *Game-based Learning*

O termo ***Game-based Learning*** refere-se ao uso de videojogos já existentes para suportar o ensino e a aprendizagem (Prensky, 2007). Um dos exemplos da aplicação deste método é o uso do videojogo criado pela empresa Blizzard, *World of Warcraft*, para ensinar matemática aos alunos na escola (Sheehy, 2015). Baseia-se em diversos princípios tais como:

- **Motivação intrínseca** – O ato de jogar é voluntário e autónomo.
- **Aprender através da diversão e do prazer**.
- **Autenticidade e contextualização** – os objetivos devem ser claros em relação à aprendizagem, rejeitando a hipótese de aprendizagem abstrata.
- **Autonomia e autossuficiência** – uso das paixões e interesses como guia para uma especialização.
- **Aprendizagem por experimentação** – aprende aplicando conhecimento na prática.

Esta metodologia apresenta diversas vantagens que a tornam bastante interessante para os alunos, tais como, o uso da paixão dos alunos como via para aprender. Isto permite uma competição saudável, visto que faz parte da natureza do jogo em que está a ser implementado o método *GBL* (*Game-Based Learning*). Assim, desenvolve-se habilidades em diversas áreas distintas, dependendo do jogo em que esteja a ser aplicado.

Devido à natureza do jogo, existem alguns problemas que devem ser ponderados quando se deseja aplicar esta metodologia, nomeadamente, a necessidade de usar um instrutor que esteja familiarizado com o jogo, conhecendo-o detalhadamente, de maneira a que este consiga controlar e canalizar os alunos a aprender. Por vezes pode existir o problema de se aconselhar o uso de um jogo e não definir quais os objetivos ou as rotas que os alunos devem tomar, podendo provocar uma diminuição do uso do jogo para aprendizagem e aumentando o seu uso para diversão.

2.3 Motor de Jogo

O conceito de motor de jogo provou ser difícil de definir, variando de programador para programador. De acordo com o artigo "*History and comparative study of modern game engines*" (Paul, Goon, & Bhattacharya, 2012), na forma mais simplista, um motor de jogo ou *game engine* em inglês, é uma plataforma que se encarrega de diversas tarefas que são comuns em todos os jogos, como *rendering*, cálculo e implementação de mecânicas físicas consoante o *input* recebido. O seu propósito é cobrir formas mais genéricas do jogo de maneira a que os próprios programadores se foquem em áreas de maior impacto, como otimizações em diversas secções do jogo de maneira a evitar quedas de fps (*frames per second*), melhorar a fluidez e mecânicas do jogo, entre outros. As vantagens do uso de motores de jogo são:

- É reutilizável;
- O número de modificações necessárias a fazer no motor de jogo, normalmente são mínimas;
- Permite uma mudança de foco de componentes mais genéricas para componentes mais específicas do jogo;
- Diminui o tempo de desenvolvimento médio de um jogo;
- Pode ser licenciado de maneira a obter-se retorno financeiro.

Contudo, segundo a definição, deveria ser possível construir qualquer jogo usando apenas um único motor de jogo. No entanto, a realidade é que a maioria dos motores são construídos cuidadosamente de maneira a funcionarem com um jogo em específico ou um género de jogo específico. As vantagens adquiridas desta abordagem são um melhor desempenho e qualidade superior. Por norma, motores mais genéricos apresentam problemas em otimização e desempenho quando comparados com motores mais específicos para um determinado género de jogo. O resultado negativo desta situação é a limitação da capacidade do programador para criar um jogo mais personalizado. Algumas das principais razões dos problemas referidos anteriormente recaem sobre o conceito de eficiência, limitação de *hardware* usado e o tipo de software implementado, ou seja, constantemente terão de ser feitos compromissos que influenciam diretamente o desempenho do jogo. Por exemplo, um motor de *rendering* desenhado para tratar de ambientes interiores, não vai apresentar uma boa performance a executar a mesma função em ambientes exteriores. Um problema bastante relevante nestes motores é a incapacidade de controlar algumas funcionalidades que lhes são implementadas, assim como adicionar novas (Gomes, 2015).

Uma das linguagens que é atualmente usada para criar jogos de *browser* é o HTML5. Um exemplo prático do seu uso é o jogo *Farmville Express* criado pela empresa Zynga (Zynga, 2015). Os jogos feitos para *browser* estão a ganhar popularidade todos os dias.Por exemplo, a empresa

Goo Technologies concluiu através de um inquérito que 52% da população Americana joga jogos de browser. As justificações apresentadas foram bastante variadas sobressaindo-se as seguintes:

- É mais barato que comprar uma consola;
- Não existe a necessidade de instalar o jogo para jogar;
- Existe uma maior liberdade de dispositivos que aceitam o jogo;
- O jogo pode ser iniciado num dispositivo e ser finalizado noutro;
- Permite jogar durante o trabalho (GooTechnologies, 2015).

Segundo as afirmações do fundador/*CEO* da companhia BoosterMedia, Laurens Rutten que afirma, "HTML5 não vai substituir os jogos "nativos", mas é de sublinhar que este se está a tornar uma tecnologia alternativa com os seus benefícios próprios…HTML5 pode ser desenvolvido para *smartphones, tablets, PCS, smart-TVS, in-car entretainment*, ou até mesmo *wearable technology* que poderá ser usada no futuro" (Rutten, 2014).

Tendo em conta todos os dados anteriores e os benefícios que a linguagem HTML5 apresenta, muitos programadores iniciaram a criação dos seus próprios motores de jogos com base nesta linguagem. Após alguma pesquisa, sentiu-se a necessidade de nomear e explicitar alguns dos motores de jogo HTML5 que existem e que se destacaram devido à sua importância atual.

2.4 Motores de Jogo HTML 5

Neste subcapítulo irão ser nomeados diversos (motores de jogo usados para construir jogos em HTML5. A pesquisa demonstrou uma grande variedade destes que usam HTML5, como tal, decidiu-se nomear apenas os cinco motores mais populares para a comunidade de programadores de jogos para HTML5.

2.4.1 *Construct 2*

O *Construct 2* é um editor de jogos baseado em HTML5. Apoia-se sobre uma interface bastante simples que permite ao utilizador criar jogos apenas através de *drag and drop* e usando a interface para fazer ajustes em relação aos comportamentos desejados, através do uso de eventos e um conjunto de comportamentos previamente definidos. Esta ferramenta suporta a ideia de criação de jogos sem a necessidade de se programar qualquer linha de código, tornando esta ferramenta bastante indicada para iniciantes sem qualquer conhecimento de programação prévia.

A ferramenta *Construct 2* permite:

- Exportar projetos para diversas plataformas, tais como, *Windows, Android, iOS, Blackberry, Windows Phone 8.0, Tizen*, HTML5, entre outras;
- Construir um jogo sem conhecimento de programação;
- Dar uso a um motor de física já construído sem ter de se proceder à construção de funções e fórmulas por parte do utilizador;
- É expansível, visto que suporta o uso de *plugins* feitos por qualquer programador, com o objetivo de adicionar funcionalidades ao *Construct 2* (Subagio, 2014).

Embora pareça uma ferramenta bastante apelativa a qualquer pessoa, possuí alguns defeitos. É necessário o uso de *plugins* para adicionar funcionalidades personalizadas e é necessário pagar para usufruir a versão completa do *software*, ou seja, a versão gratuita está limitada em funcionalidades e não permite ao utilizador exportar o projeto para *Windows, Android* ou qualquer outra plataforma (Scirra, 2015).

2.4.2 *ImpactJs*

ImpactJs é um dos primeiros motores de jogo profissionais que aproveitam o elemento *Canvas* do HTML5 para criar jogos 2D em *Javascript*. Jogos criados através do uso deste, tal como a maioria dos jogos HTML5, vão correr em "qualquer *browser* moderno", afirma Darius Kazemi em 2011, membro da equipa responsável pelo desenvolvimento do motor de jogo. O motor possui diversas funcionalidades, tais como o *Weltmeister Level Editor* que permite a criação de mapas usando diversos materiais fornecidos pelo *site* oficial, suporte para dispositivos móveis que dá uso a uma função de autodetecção do dispositivo onde está a ser rodada a aplicação, de modo a personalizar as configurações da aplicação, com o objetivo de aumentar o desempenho da mesma. Segundo Darius Kazemi, a linguagem *javascript* não "percebe" o conceito de *includes*, então a equipa acrescentou um sistema de módulos que permite a organização do código. Outra funcionalidade que é bastante vantajosa neste *engine* é o uso de um script que transforma o jogo criado num ficheiro JS juntamente com uma série de ficheiros mais pequenos que tem como objetivo diminuir o número de *requests* e o tempo de *load* do jogo (Szablewski, 2011).

A única desvantagem encontrada neste motor é o fato de ser pago, com um custo de 99€. Tendo em conta a grande variedade de motores de jogo HTML5 open-source, a não ser que o programador necessite de produzir um produto comercial de alta qualidade, o preço não justifica a vantagem que se irá ganhar.

2.4.3 *CreateJS*

De acordo com o *site* oficial, "*CreateJs* é um conjunto de ferramentas e bibliotecas modulares que trabalham em conjunto ou independentemente para criar conteúdo enriquecedor usando como base HTML5" (Skinner, Mcnie, & Gorgichuk, 2015). Este conjunto de ferramentas está dividido em 4 bibliotecas distintas:

- *EaselJS;*
- *TweenJS;*
- *SoundJS;*
- *PreloadJS;*

EaselJS

O EaselJS é uma biblioteca *Javascript* que providencia ao utilizador a capacidade de desenhar no elemento *Canvas* do HTML 5. Toda a interação e o *rendering* gráfico será realizado com o uso desta biblioteca. Esta também capacita o utilizador com:

- A possibilidade de construir uma lista de hierarquia de objetos a colocar no ecrã.
- Um modelo de interação e um conjunto de classes pré-feitas para facilitar o uso de grafismo 2D com o *Canvas*.

Lamentavelmente não possibilita o uso de *WebGL*, sendo esta uma das limitações presentes nesta biblioteca (Manderscheid, 2014).

TweenJS

A biblioteca *TweenJS* permite animar facilmente os objetos que estão a ser mostrados pela biblioteca *EaselJS*. Segundo a documentação oficial, esta biblioteca fornece uma interface que permite animar diversos tipos de objetos, incluindo objetos com propriedades CSS e objetos com propriedades numéricas. Também dá a capacidade ao utilizador de fazer diversas animações e ações em cadeia, de modo, a criar sequências mais complexas e personalizadas (Skinner, Mcnie, & Gorgichuk, 2015).

SoundsJS

Embora o fator visual tenha bastante peso em qualquer jogo, a componente auditiva também merece grande destaque. Esta tem uma responsabilidade de peso igual ao superior em proporcionar imersão para o utilizador comparando com a componente visual. Segundo a documentação, a biblioteca *SoundJS* foi desenhada com o objetivo de fornecer ao programador a capacidade de reproduzir áudio na *Web*. Funciona através de *plugins* que se abstraem da implementação de áudio, permitindo assim a qualquer plataforma reproduzir áudio sem qualquer conhecimento prévio dos mecanismos necessários para reproduzir sons (Skinner, Mcnie, & Gorgichuk, 2015).

PreloadJS

A última biblioteca (PreloadJS) não possui muita informação, quer no *site*, quer na sua documentação. No entanto a sua função é fornecer ao programador uma maneira de fazer *pre-load* de conteúdo para depois ser usado nas aplicações HTML.

2.4.4 *PixiJs*

Vivemos numa época onde tempo é um fator bastante importante, por muito que o ser humano queira, tempo é algo que não se consegue recuperar. Sendo este um fator importante para o ser humano, também é um fator decisivo e de bastante peso nas escolhas de *software* e *hardware*. O motor *PixiJs* tem como objetivo principal ser um motor de *rendering* de *sprites* 2D, sendo este bastante rápido segundo o *site* oficial e o autor Rex van der Spuy (Spuy, 2015). Este motor de jogos permite animar, mostrar e gerir gráficos, assim como os juntar em grupos de maneira a dar mais capacidade ao programador de personalizar todos os esquemas que desejar.

Este também está equipado com *scene graph* que permite criar hierarquia e o chamado *nested sprites* (*sprites* dentro de *sprites*). Existe também a possibilidade de adicionar eventos consoante o *input* recebido, suporta o *rendering* de texto, possui diversos filtros para as imagens ou *sprites* que forem acrescentadas/os e é multiplataforma. A desvantagem deste motor é o facto de ser bastante específico, visto que foi criado para ser um *renderer WebGL* 2D, fornecendo poucas opções em termos de áudio, mecânicas físicas, *preload*, entre outros. No entanto, devido à sua grande capacidade e velocidade de *rendering*, muitos motores de jogos usam este motor para tratar desta componente, um exemplo deste caso é o *Phaser.Js*.

2.4.5 *Phaser.Js*

Phaser é um motor *open-source* de HTML5 que permite criar jogos que vão ser interpretados por um *web browser*. A criação deste, embora já existam muitos no mercado, foi devido a uma combinação de necessidade, frustração e oportunidade, segundo o seu criador (Davey, 2015). . Este motor é constituído por componentes de outros motores, cada um especializado na sua área. O motor responsável pelo *rendering*, é o motor Pixi.JS, um dos motores mais rápidos nessa função em 2D, ao qual será dada mais importância adiante.

O *Phaser.JS* possuí as seguintes funcionalidades:

- Capacidade de usar *WebGL* e *Canvas*;
- Possuí um *Pre-Loader*;
- Tem 3 motores de mecânicas físicas distintos;
- Permite adicionar e manipular *sprite* (objeto gráfico);
- Reconhece *inputs* de computadores e de dispositivos móveis.
- Funciona com os *browsers* de dispositivos móveis.
- Deteta e ajusta as configurações de áudio automaticamente consoante o *browser* a ser usado.
- Permite adicionar sons de maneira simples.
- Possibilita a animação de *sprites*, pelo uso de *tweens*(processo que gera frames intermédias de uma imagem de maneira a simular o movimento de uma imagem) ou a partir de uma animação pré-feita.

Este motor demonstra grande potencial e apresenta bastantes vantagens para o programador que decida usá-lo para futuros projetos. Uma das maiores vantagens do *Phaser.JS* é a quantidade de *updates* que recebe, este motor está em constante evolução. A sua comunidade é bastante grande, dando ao utilizador suporte constante por parte de programadores da comunidade, assim como, programadores do *Phaser*. Pela mesma razão, *bugs* presentes neste são rapidamente detetados, visto existir uma grande área de *feedback*. É bastante acolhedor para quem deseja iniciar a sua viagem na programação de jogos, exatamente porque a comunidade dispõe de um conjunto de guias e exemplos práticos com código que pode ser compilado e executado *online*.

As desvantagens detetadas neste motor são, a necessidade de um conhecimento prévio de programação e a documentação possuir certos detalhes que poderiam estar mais explícitos.

2.4.6 Sumário

Tabela 2.2 Tabela de Comparação entre os diferentes Motores de Jogo HTML5

Características\Motores de Jogo	***Construct 2***	***ImpactJS***	***CreateJS***	***PhaserJS***	***PixiJS***
Programação	Não Precisa	Javascript	Javascript	Javascript	Javascript
Suporte WebGL	Sim	Sim	Não	Sim	Sim
Documentação	Boa	Muito Boa	Boa	Muito Boa	Boa
Personalizável**	Não	Sim	Sim	Sim	Sim
Custos	Versão free muito limitada	99 €	Gratuito	Gratuito*	Gratuito

*Existem *plugins* pagos, mas o conteúdo pode ser replicado com programação

**Permite alteração e criação de funções e funcionalidades.

Em suma, analisando-se para a Tabela 2.2, destacam-se os motores *ImpactJS* e *PhaserJS* na documentação. O *CreateJS* e o *PixiJS* destacam-se no campo dos custos, onde se apresentam como motores 100% *open-source*, sem qualquer tipo de custos acrescidos seja por funcionalidades extras ou *plugins*. Finalizando com o motor *Construct 2* que se sobressai dos restantes devido à simplicidade do mesmo, não sendo necessário obter qualquer tipo de conhecimento na área da programação para puder usufruir deste. Após serem avaliadas as diversas vantagens e desvantagens que todos os motores de jogo apresentam, decidiu-se escolher o motor de jogo Phaser.JS como ferramenta a adoptar nesta dissertação, visto que se destaca dos restantes por ser uma ferramenta Open-source que apresenta uma grande quantidade e qualidade da documentação aliada ao suporte fornecido online.

2.5 *Web Services*

2.5.1 Definição de *Web Services*

Atualmente, o requisito básico de qualquer empresa de serviços é fornecer serviços, sejam eles vendas, compras, etc. Sendo a comunicação um fator chave para o sucesso de uma empresa, viu-se a necessidade de o reduzir, de modo a aumentar o desempenho e os lucros da mesma. Verificou-se que muitos serviços poderiam ser automatizados, diminuindo assim drasticamente o tempo de demora dos processos e evitando o tempo de comunicação para processos menos importantes, aumentando assim o foco e a redistribuição do tempo para tarefas e processos mais urgentes. Os Web Services são criados para resolver este problema, podendo criar aplicações que são serviços na internet (Pamplona, 2015).

O conceito de Web Service, segundo Ethan Cerami é qualquer serviço disponível na internet que usa o sistema *standard* de mensagens XML para comunicar e não depende de qualquer sistema operativo ou linguagem de programação. Este também afirma que existem diversas alternativas ao sistema de mensagens XML, entre elas:

- XML-RPC, que se baseia no uso de XML para se fazer procedimentos remotos, daí o nome XML Remote Procedure Calls;
- SOAP – Protocolo Simples de Acesso a Objetos;
- O uso de HTTP Get/Post passando documentos XML como argumentos (Cerami, 2002).

Contudo, a definição selecionada para descrever o conceito de Web Service nesta dissertação é dada por James Snell, em que afirma que este é uma interface acessível pela rede para o funcionamento de uma aplicação, usando como base tecnologias de internet *standard*.

Em suma, se uma aplicação pode ser acedida através da internet perante um conjunto de protocolos como, HTML, XML, STML, *Jabber*, entre outros, então é um *Web Service* (Snell, Tidwell, & Kulchenko, 2001).

2.5.2 Definição de Serviços SOAP

Tendo em conta o conceito de *Web Services* nomeado anteriormente, irá então ser definido a uma das diversas aplicações práticas deste conceito.

SOAP é o acrónimo para *Simple Object Acess Protocol*, ou, traduzindo para português, Protocolo Simples de Acesso a Objetos. Este protocolo define uma mensagem baseada em XML para a informação ser transportada, juntamente com um conjunto de regras para a aplicação que terá a função de fazer a tradução e um conjunto de especificações típicas de tipos de dados em representações XML. SOAP é definido por 3 componentes importantes:

- Especificação do envelope SOAP;
- Regras de encriptação de dados;
- Convenções RPC (Cerami, 2002).

Um exemplo da aplicação de um serviço SOAP e o seu funcionamento pode ser observado na figura 2.2 :

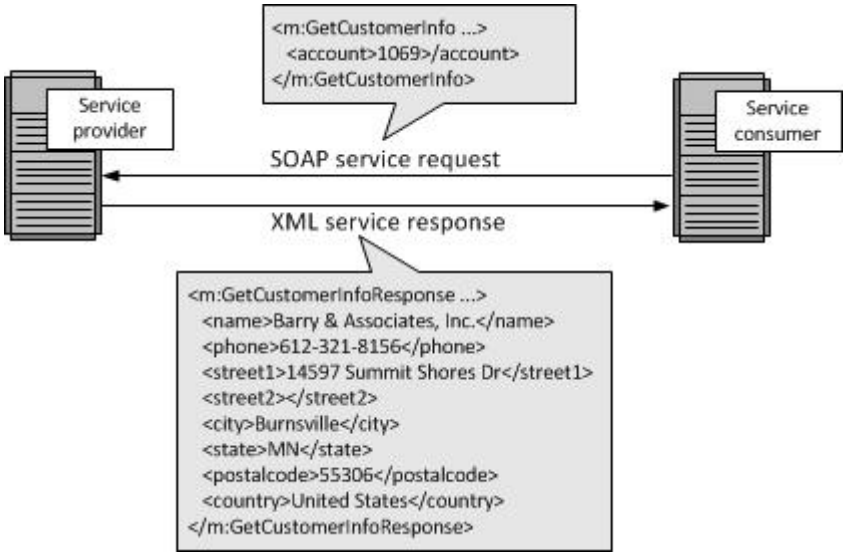


Figura 2.8 Exemplo de um serviço SOAP (Fonte: http://www.service-architecture.com/articles/web-services/web_services_explained.html).

Especificação do envelope SOAP

Um envelope de SOAP é baseado em XML e define um conjunto de regras específicas para encapsular dados a serem transferidos entre computadores. Também são incluídos dados específicos da aplicação, tais como nome do método que irá ser invocado, parâmetros de entrada e valores devolvidos pelo mesmo. Pode também conter informação sobre quem deve processar o conteúdo do envelope e, em caso de erro, como desencriptar mensagens de erro (Cerami, 2002).

Regras de Encriptação de Dados

Um dos propósitos deste protocolo é a capacidade de trocar dados de maneira segura, como tal, é necessário proceder-se à encriptação da informação de maneira a manter a sua integridade. É neste campo que as regras de encriptação atuam. Inicialmente, para existir uma troca de dados, é necessário que ambos os computadores aceitem as regras para encriptar os tipos de dados em causa. Uma vez aceites, o protocolo SOAP inclui o seu conjunto de regras e convenções que ambos os computadores irão usar para codificar e descodificar mensagens (Snell, Tidwell, & Kulchenko, 2001).

Convenções RPC

SOAP pode ser usado por um leque extenso de sistemas de mensagens, incluindo comunicação unidirecional e bidirecional. No caso da comunicação bidirecional, o protocolo SOAP define uma convenção simples para representar respostas e chamadas do procedimento remoto. Permite-se assim à aplicação do lado do cliente, especificar um nome de um método que pode ser acedido remotamente, incluindo informação sobre qualquer número de parâmetros (Snell, Tidwell, & Kulchenko, 2001).

2.5.3 Arquitetura REST

RESTful tem como base o acrónimo *REST* que significa *Representational State Transfer*, ou, traduzindo para português, Transferência de Estado Representacional. É um estilo arquitetural que tem como objetivo principal, especificar diversas restrições que se podem traduzir em diversas melhorias quando aplicadas aos serviços Web (Web Services), nomeadamente o desempenho, a escalabilidade, entre outros (Oracle, 2015).

Segundo o Dr. Roy Fielding, criador do conceito de *REST* para a sua dissertação de doutoramento, existem 6 restrições que constituem esta arquitetura:

- Cliente-Servidor;
- *Stateless*;
- Cache;
- Interface Uniforme;
- Sistema por Camadas;
- *Code on Demand*.

Deve salientar-se que estas restrições servem apenas de guias, não são mandatárias, o que explica a existência de serviços web que seguem apenas algumas delas.

Cliente-Servidor

A primeira restrição propõe que o servidor e o cliente sejam construídos de maneira a que ambos sejam independentes, ganhando assim a capacidade de evoluírem separadamente. Exemplificando: a pessoa responsável pela aplicação ou serviço terá a capacidade de alterar o cliente sem influenciar de maneira nenhuma os dados, bases de dados, estruturas de dados, etc, pertencentes ao servidor. A lógica mantêm-se inversamente, ou seja, deve-se dar a capacidade de se conseguir alterar a base de dados ou as estruturas presentes na componente do servidor sem existir qualquer tipo de modificação na componente do cliente. As vantagens adquiridas com a implementação desta restrição são:

- As aplicações podem escalar separadamente;
- Independência entre ambos os conceitos;
- Maior organização;
- Facilita algumas alterações (Fielding, 2000).

Stateless

O nome desta restrição traduzindo para português, significa nos termos mais simples “sem estado”. Esta restrição dita que o servidor web não necessita, ou não deve, memorizar o estado das aplicações dos clientes. Isto é, cada cliente deve incluir toda a informação que considere relevante para cada interação que este tenha com o servidor web (Massé, 2012).

Em termos técnicos, a restrição *Stateless* define que todas as *calls* (chamadas ao servidor) não devem ser dependentes umas das outras e cada uma delas deve ter informação (dados) necessária para se auto completar com sucesso. Uma aplicação que use a arquitetura REST não deveria ter a necessidade de guardar informação no servidor para decidir o que deve fazer numa determinada *call* devendo-se apenas confiar na informação que lhe é fornecida por esta. Em suma,

a informação de qualquer estado deve ser guardada no cliente e não no servidor. As vantagens obtidas ao seguir esta restrição são:

- Redução do uso de memória;
- Aumento da fiabilidade da aplicação;
- Manutenção da escalabilidade da aplicação.

Cache

Segundo Mark Massé, *caching* é uma das restrições mais importantes do *REST*. A sua função é capacitar o servidor web de declarar a quantidade de cache de cada resposta de dados. Ou seja, esta restrição reforça a ideia de que o cliente deve guardar informação do seu lado com o objetivo de reduzir interação com o servidor. Massé também reforça que "*Caching* de dados pode ajudar a reduzir a latência *client-perceived*, aumentar de um modo geral a disponibilidade e a fiabilidade de uma aplicação e um aumento no controlo da carga do servidor web" (Massé, 2012).

Interface Uniforme

Todo o tipo de comunicação presente no mundo, baseia-se no pressuposto de que é necessário usar determinada linguagem para se conseguir transmitir informação. Exemplificando: um indivíduo que viaja para uma cidade estrangeira, para conseguir comunicar com residentes locais, terá de saber a sua linguagem, ou não conseguirá comunicar verbalmente com eles. Caso não consiga, terá de encontrar um individuo que partilhe a mesma linguagem que ele ou terá de tentar outros tipos de comunicação, como por exemplo, comunicação gestual.

No caso da Web, a situação é bastante semelhante. Para existir comunicação, ambas as partes têm de seguir certos *standards*. Caso contrário, qualquer desvio destes *standards* pode fazer com que a comunicação cesse por completo. Isto é, é necessária uma uniformidade nas interfaces de todos os componentes web, para manter a integridade da comunicação da Web. Segundo Roy Fielding, as componentes Web devem inter-operar consistentemente, seguindo 4 restrições da interface uniforme:

- Identificação de recursos;
- Manipulação de recursos através de representações;
- Mensagens autodescritivas (*Self-descriptive*);
- O uso de *Hypermedia* como motor do estado da aplicação (HATEOAS) (Fielding, 2000).

Identificação de Recursos

Em suma, a identificação de recursos explica que tudo o que seja construído baseado no conceito Web e que pode ser acedido por um identificador único, tal como o URI, sendo este considerado um recurso (Fielding, 2000).

Exemplos:

- Uma criação de um cliente - **http://www.exemplo.com/cliente**
- Uma leitura de um cliente com o id 450 - **http/www.exemplo.com/clientes/450**
- Uma criação de um produto - **http://www.exemplo.com/produto**

Manipulação de recursos através de representações

Tem de ser dada a capacidade ao mesmo recurso de ser representado de diversas maneiras para diversos clientes. O exemplo dado por Mark Massé é o seguinte, "um documento pode ser representado por HTML para um *browser* da *Web* e um *JSON* para um programa automatizado" (Massé, 2012).

Mensagens autodescritivas

Quando o utilizador recebe ou envia mensagens, por vezes pode ser necessário transmitir o estado em que um determinado recurso se encontra. Como tal, a restrição das mensagens autodescritivas vem auxiliar nessa situação. Uma mensagem de *request* ou *response* pode conter uma representação do estado em que um recurso deve estar. Por exemplo, se uma estudante deseja inscrever-se numa determinada cadeira online, cabe ao servidor aprovar ou rejeitar a sua inscrição, a aluna pode, eventualmente, não ter créditos suficientes.

HATEOAS

HATEOAS é o acrónimo de *Hypermedia as the Engine of Application State* e tem como objectivo, "a partir de um determinado recurso ser possível descobrir a localização de todos os recursos que lhe estão associados" (Sousa, 2015).

Ao analisar-se a estrutura de um recurso, pode-se verificar que existe uma secção que contém um nome chamado *metadata*. Essa secção será comparável ao que conhecemos como Lista telefónica. É uma secção que contém um conjunto de representações de recursos juntamente com os respetivos *links*. Posteriormente a aplicação vai usar os *links* para se ligar aos recursos assim que achar necessário (Doglio, 2015).

A vantagem do uso de HATEOAS é que o cliente não necessita de conhecimento prévio de como interagir ou usar um determinado recurso, podendo-se assim fazer modificações aos recursos, sem qualquer implicação para o cliente.

Sistema por Camadas

Resumidamente, sistema por camadas no seu conceito mais básico é um sistema constituído por diversas camadas, em que cada uma delas tem uma determinada função, tal como uma determinada responsabilidade. Esta restrição traz como vantagem, uma simplificação dos componentes visto que as camadas apenas permitem o uso da camada imediatamente abaixo e a comunicação com a camada imediatamente acima. Também pode ser verificado que devido ao uso desta restrição aumenta-se a possibilidade de escalamento da aplicação, visto que pode sempre acrescentar-se mais camadas para aumentar funcionalidades (Stowe, 2015).

Code-on-Demand

Como foi dito anteriormente, esta restrição é completamente opcional, visto que não se enquadra em todos os problemas que uma API REST se foca. *Code-on-Demand* é uma restrição que num conceito mais simplista, dá a capacidade ao cliente de fazer *download* de uma porção de código fornecido pelo servidor e de o executar (Doglio, 2015). A ideia por detrás desta restrição seria transformar uma aplicação em algo do futuro, uma aplicação "smart". No entanto, podem-se verificar problemas de segurança uma vez aplicada (Stowe, 2015).

2.5.4 RESTful Web services

Os serviços RESTful são serviços web que são baseados na arquitetura REST. Os serviços RESTful aproveitam diversas restrições estipuladas nessa arquitetura visto possuir propriedades que melhoram o desempenho, escalabilidade, personalização e permitem um melhor funcionamento na Internet. As restrições que normalmente são usadas e se destacam pela simplicidade e velocidadde são:

- **Identificação de Recursos –** Um serviço RESTful expõe um conjunto de recursos que identifica todos os possíveis caminhos ou recursos necessários para a interação com os clientes. Tal como foi dito anteriormente no capítulo 2.5.3, na secção de Identificação de recursos, os recursos são identificados por URIs.
- **Interface Uniforme –** Os recursos são manipulados por um conjunto de operações conhecidas, PUT, GET, POST e Delete. Em que o PUT é responsável por criar um nbovo recursos que pode ser eliminado através do uso do Delete. A operação GET recolhe o estado actual do recurso e o POST transfere o novo estado para o recurso.
- **Mensagens auto-explicativas** – Acrescentando ao que foi explicado no capítulo 2.5.3 na secção das Mensagens Auto-explicativas, o conteudo dos recursos pode ser acedido por um conjunto de formatos, tais como HTML, XML, plain text (texto), PDF, JPEG, entre outros (Oracle, 2013).
- **Stateless –** Explicado anteriormente no capítulo 2.5.3 na secção Stateless.

Um exemplo da aplicação de serviços RESTful pode ser visto na figura 2.3:

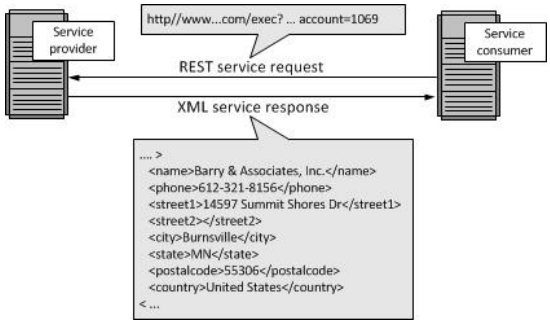


Figura 2.9 Exemplo de RESTful Web Services (Fonte: http://www.service-architecture.com/articles/web-services/web_services_explained.html).

2.6 MVC e NodeJs

2.6.1 MVC

MVC vem do acrónimo Model View Controller (Modelo Visão Controlador) e é um padrão de arquitetura de software que tem como objectivo separar uma aplicação em 3 componentes lógicos:

- Modelo
- Visão
- Controlador

Cada um destes componentes está construido para cuidar de aspectos especificos de desenvolvimento de uma aplicação. Normalmente este tipo de padrão é usado para criar projectos escaláveis e extensiveis. Um diagrama representativo de um MVC pode ser observado na figura 2.10.

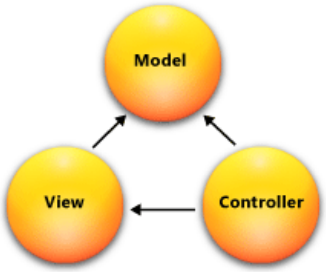


Figura 2.10 Representação do MVC

Modelo

O modelo é uma das partes da aplicação que está responsável pelos dados e pela lógica de negócio. Este pode representar os dados que estão a ser transferidos entre a Visão e o controlador ou qualquer outro tipo de lógica de negócio relacionada com os dados. Por exemplo: Um objecto cliente pode recolher informação do base de dados cliente, manipula-la e modifica-la ou usa-la para ser visualizada.

Visão

A Visão é a componente usada para cuidar de toda a lógica UI da Aplicação. Normalmente a UI é criada a partir dos dados do Modelo. Um exemplo de uma Visão do modelo do Cliente seria uma forma de apresentação, como uma página HTML5 com todos os componentes de uma UI tais como, caixas de texto, botões, etc, com que o utilizador possa interagir directamente.

Controlador

Os controladores são componentes que agem como uma interface entre o Modelo e a Visão para cuidar de toda a interação com o utilizador, trabalhar com o modelo e para seleccionar a Visão que deve ser disponibilizada.

2.6.2 Node JS

Node.Js é uma plataforma construída sobre o motor Javascript do Google Chrome e é usado para desenvolver aplicações rápidas e escaláveis, recorrendo a um modelo de I/O (*Input/Output*) regido por eventos não bloqueantes (Node.js, 2015). Todas as características nomeadas anteriormente tornam o Node.js leve e eficiente para aplicações intensivas como sites de streaming, aplicações single-page, entre outras. É open source, gratuito e usado por diversos programadores espalhados por todo o mundo.

Esta plataforma permite ao programador, criar uma componente cliente e servidor sem a necessidade de saber mais que uma linguagem visto que a linguagem Javascript pode ser usada em ambas as partes. Deve sublinhar-se contudo que, o Node.js requer alguma experiência e não é adequado para iniciantes devido à forma como a programação de eventos funciona.

É de notar que o "Node.js não depende de multithreading na execução de processos concorrentes, na realidade, deve-se interpretar o Node.js como uma plataforma de desenvolvimento de servidores Web que dá capacidade ao programador de contruir sistemas altamente escaláveis" (Sousa, 2015). Contudo, sem demasiada complexidade de gestão que um sistema multithreading possa apresentar. Tendo em conta, que este opera apenas com uma simples thread, existe quem crie várias instâncias do Node.js para ocupar todos os cores que o processador de um servidor tenha.

3

Proposta

Neste capítulo, inicialmente, irá ser feita uma análise em relação ao problema desta dissertação, seguindo-se de um conjunto de especificações que são os requisitos necessários para o funcionamento de uma possível solução. Finalizando-se com a apresentação de um modelo que deverá cobrir todos os requisitos pedidos juntamente com uma solução para o problema analisado inicialmente.

3.1 Análise do Problema

Como foi dito anteriormente, todos os anos nasce 1 em cada 800 crianças com síndrome de *Down*, afetando um total de 15.000 pessoas segundo os dados da associação Pais 21 (Pais21, 2016). Estas crianças apresentam dificuldade no desenvolvimento auditivo, visual, linguagem e fala, podendo-se verificar um défice de memória auditiva a curto prazo juntamente com reduzida capacidade de atenção e dificuldade de retenção e consolidação da informação (Apatris21, 23).

Perante todas as dificuldades apresentadas, sente-se a necessidade de investir em projetos de modo a auxiliar a educação e integração destas crianças na sociedade. Tendo por base o auxilio na educação e integração, surge o projeto que deu origem a esta dissertação, em que se procura desenvolver uma aplicação que cubra várias áreas onde existem problemas associados ao ensino da linguagem e da consciência fonológica. Este tipo de aplicações não é comum por terem de ser específicas para o objetivo pretendido.

O desenvolvimento destas aplicações tem implícito um conjunto de desafios, tais como, a necessidade de provocar a mínima ou nenhuma distração ou desmotivação no utilizador, e capacitar a aplicação de produzir o objetivo desejado, e restrições, tal como, a não introdução de funcionalidades para que os objetivos delineados sejam atingidos. A criança deve-se identificar com o personagem principal e este deve ser usado como reforço positivo para a motivar a continuar a jogar. A simplicidade e fácil manuseamento, quer para o jogador, quer para a pessoa que a acompanha, é uma característica obrigatória.

3.2 Especificações do Sistema

Foi estipulado que se iria desenvolver um jogo de puzzles que possui diversos mini-jogos cada um com o seu objectivo especifico. Este jogo terá de ser reproduzido em diversas plataformas, todas elas suportando HTML 5. Recolha de dados e criação de contas de utilizador é um factor importante de modo a possuir registos detalhados de cada jogador.

3.2.1 Requisitos Funcionais

Segundo a empresa Makesys, requisitos funcionais são os fundamentais que definem uma função do *software* ou parte dele, determinando a reação que este apresenta perante as entradas e o seu comportamento em situações específicas (Makesys, 2015). Com base no problema descrito e no conceito acima enunciado, os requisitos funcionais que foram definidos para este projeto são:

- O sistema deve englobar diversas atividades, explicando para cada uma as instruções;
- Todas as atividades desenvolvidas no sistema devem contemplar aleatoriedade da posição dos itens no ecrã;
- Todas as atividades devem ser acompanhadas de componentes auditivas e visuais
- O sistema deve permitir a gravação de dados de forma automática juntamente com a capacidade de os mostrar visualmente;
- Deve ser permitida a criação, edição e eliminação de várias contas de utilizador, bem como consultar os seus dados;
- Deve ser permitido interromper a atividade em que o utilizador está sem qualquer perda de informação.

3.2.2 Requisitos Não Funcionais

Requisitos não funcionais estão relacionados com o uso da aplicação no âmbito do desempenho, usabilidade, confiabilidade, disponibilidade, segurança e tecnologias envolvidas. Por vezes os requisitos não funcionais acabam por criar restrições aos requisitos funcionais, de acordo com a empresa Makesys (Makesys, 2015). Segundo a descrição do problema desta dissertação e, tendo em conta o conceito introdutório deste subcapítulo foram determinados os seguintes requisitos não funcionais:

- A execução desta aplicação deve ser suportada em dispositivos móveis e computadores;
- A interface deve ser autoexplicativa, intuitiva e de simples manuseamento;
- O sistema deve ser construído de maneira a garantir a escalabilidade futura, mantendo a integridade do mesmo;
- Deve ser permitido adicionar mais atividades sem influenciar as que já se encontram implementadas;
- Deve ser possível restringir acesso aos utilizadores de maneira a proteger a integridade do sistema.

3.3 Solução Proposta

3.3.1 Arquitetura da Solução

Após a análise do problema, procedeu-se à modelação de uma solução que conjugue e satisfaça todos os requisitos nomeados anteriormente nos subcapítulos dos requisitos não funcionais e funcionais (3.2.2 e 3.2.1, respetivamente) desta dissertação.

Idealizou-se uma arquitetura, em que existe uma base de dados, uma API (*Application Program Interface*) e um jogo. O uso de uma API é essencial para que haja uma comunicação entre os diversos componentes e permite que cada uma possa ser atualizada de forma independente. Com esta arquitetura pode-se ampliar as áreas de ação do jogo, criando e implementando novas atividades sem se comprometer a base de dados e vice-versa. Um diagrama de exemplo da arquitetura sugerida anteriormente pode ser observado na figura 3.1.

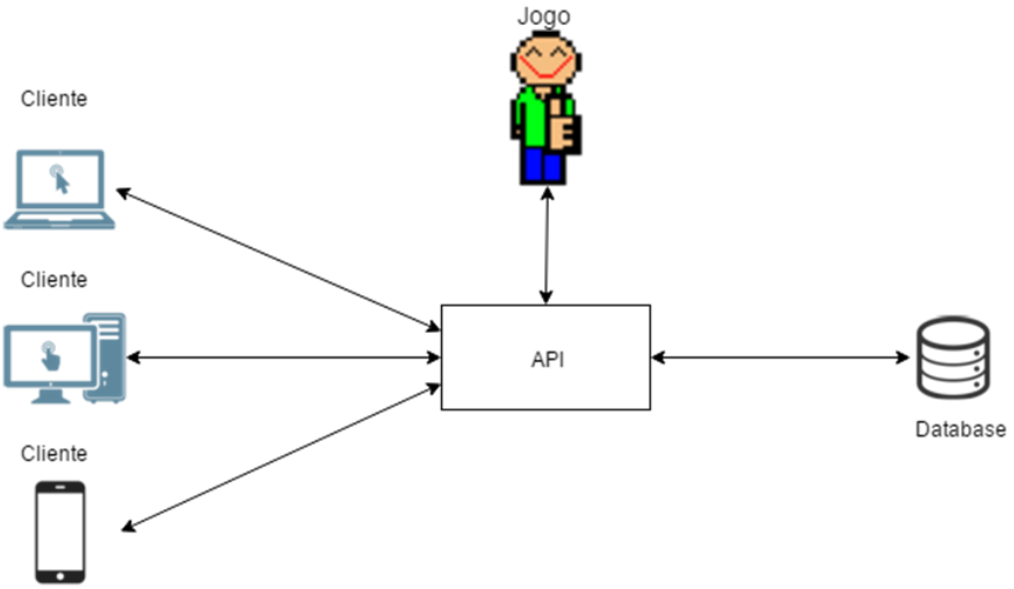


Figura 3.1 Representação da estrutura da solução

3.3.2 Estrutura do Jogo

O jogo será esquematizado da seguinte maneira: será iniciado um vídeo introdutório no seu arranque, seguindo-se um breve ecrã que irá conter uma barra dinâmica ou outra ferramenta que tenha a capacidade de demonstrar visualmente o progresso do carregamento de dados ao utilizador. Uma vez feito o carregamento (*loading*) dos dados, o menu principal deverá ser carregado, sendo que este deve conter os vários minijogos e por consequência o menu principal deve encarregar-se de permitir ao utilizador mostrar os vários jogos como prosseguir para um selecionado.

Assim que o minijogo seja escolhido deverá ser precedido por um vídeo de forma a provocar a imersão do utilizador no jogo. Quando o vídeo finalizar, será iniciado o primeiro nível, correspondente ao minijogo escolhido, da mesma maneira que está representado na figura 3.2.

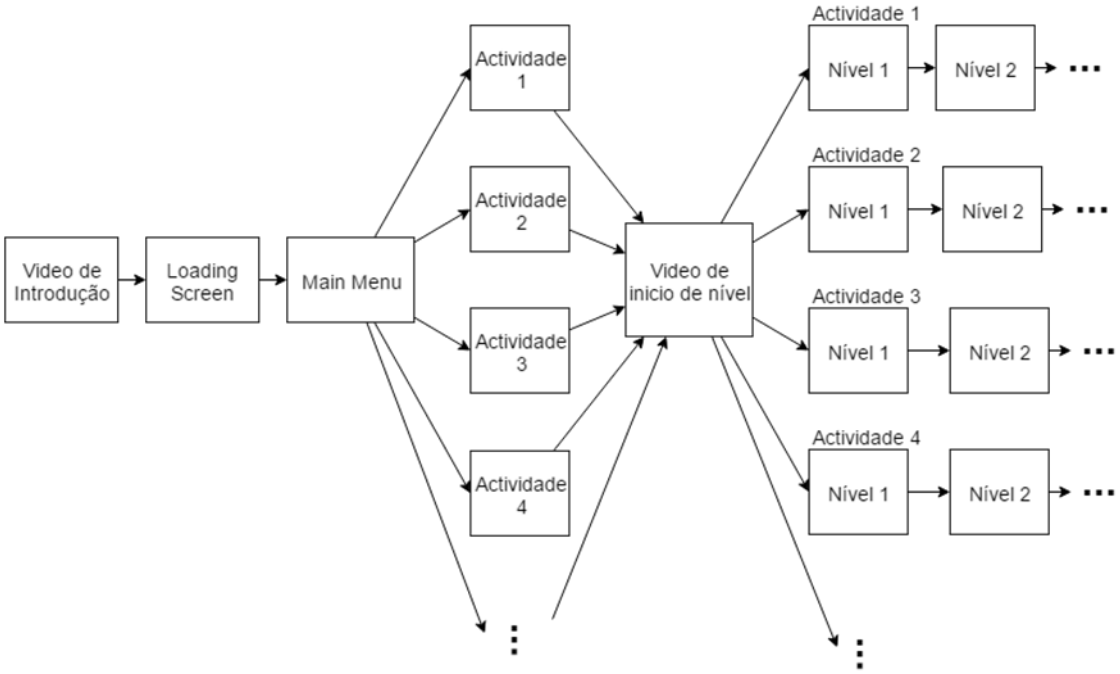


Figura 3.2 Estrutura do Jogo

3.3.3 Estrutura dos Minijogos Palavra a Palavra e Fraseando

O modelo apresentado pelo Centro Diferenças para o minijogo Palavra a Palavra é representado na figura 3.4. em que se baseia na repetição de uma palavra enquanto está a ser mostrada uma imagem. A imagem a mostrar deverá ser escolhida de forma aleatória e apresentada no centro do ecrã. Após esta ser posicionada, deverá ser apresentado um texto que identifique o conceito demonstrado pela imagem, como se pode observar na figura 3.3. Quando esta for carregada (quando o utilizador proceder ao clique da imagem), deverá ser produzido um som com a leitura do texto.

Figura 3.3 Exemplo de *Layout* no jogo Palavra a Palavra/Fraseando

A validação deste minijogo poderá ser feita de duas maneiras, deteção do som produzido pelo utilizador e acompanhamento, por parte de um assistente, para validar se o som produzido é o correto.

Para este minijogo, o Centro Diferenças propôs que sejam mostradas 5 imagens para que se evite o aborrecimento e desconcentração do utilizador durante a progressão do minijogo. A diferença entre o minijogo Palavra a Palavra e o Fraseando está na dificuldade do conteúdo apresentado. Em que, a Palavra a Palavra apenas apresenta figuras e textos com uma palavra, enquanto o Fraseando apresenta frases.

Figura 3.4 Estrutura dos mini-jogos Palavra a Palavra e Fraseando

3.3.4 Estrutura dos Minijogos Contar os Sons e Contar os Bocadinhos

Contar os sons é um minijogo que obriga o aluno a ter de realizar duas tarefas distintas: dividir uma palavra em fonemas e contar quantos fonemas existem. Quando o nível for iniciado, deverá ser escolhida uma imagem aleatoriamente, e a imagem deverá ser colocada no centro do ecrã. Posteriormente irá ser calculado o número de *smilies* que são necessários colocar, estes irão corresponder à quantidade de fonemas ou silabas que a palavra correspondente à imagem contém, como se pode verificar na figura 3.5. Quando o utilizador clicar nos *smilies* estes devem mudar de cor em que, a ordem pela qual devem ser carregados deverá ser da esquerda para a direita, de forma seguida. Além de ensinar o utilizador a contar o número de fonemas, este exercício também o força a aprender a importância da sua ordem.

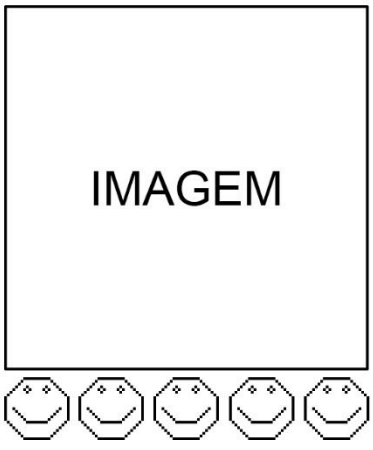


Figura 3.5 Exemplo de *Layout* do minijogo Contar os Sons

De forma a evitar que o utilizador não mecanize o método de avançar de níveis, deverão ser incluídos uns *smilies* extras de forma a despistar esta situação e estes devem ser incluídos num estado mais avançado do minijogo. Isto vai implicar que, para estes níveis, não vai ser possível uma transição automática para o nível seguinte, por ser necessária uma validação de modo a confirmar que os *smilies* carregados são os corretos, podendo esta ser realizada através de um botão. Uma vez realizada a validação, o jogo deverá enviar um feedback ao utilizador para que este saiba se acertou. Assim que o utilizador carregar no número certo de *smilies*, o nível deverá avançar automaticamente. A quantidade de turnos ou imagens que será apresentada neste minijogo corresponde a 5 turnos em qualquer nível de dificuldade. Um resumo do que foi descrito anteriormente pode ser observado na figura 3.6.

As diferenças entre o Contar os Sons e Contar os Bocadinhos irão ser apenas na maneira como é dividida a palavra que está a ser mostrada, visto que, o minijogo Contar os Bocadinhos divide a palavra por sílabas. No entanto as mecânicas base de apresentação da imagem e dos *smilies* deverão ser idênticas ao Contar os Sons.

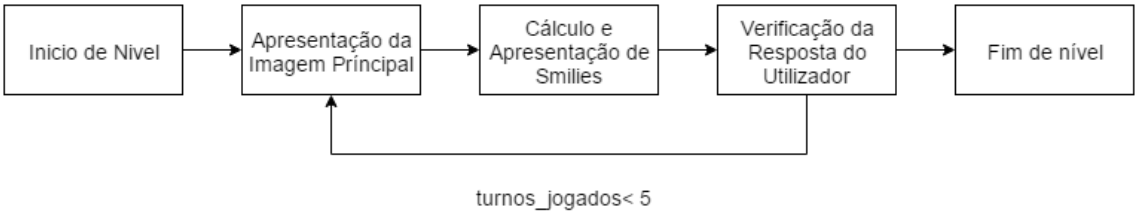


Figura 3.6 Estrutura dos minijogos Contar os Sons e Contar os Bocadinhos

3.3.5 Estrutura dos Minijogos Palavras a Rimar e Sons Iniciais

Nenhum dos minijogos atrás mencionados explora o uso de mais que um som ao mesmo tempo. Os minijogos Palavras a Rimar e Sons Iniciais baseiam-se nesse objetivo. Uma vez concluído o carregamento do nível, deverá ser apresentado ao utilizador uma imagem aleatória que vai servir de referência para o resto do minijogo. Após uns instantes deverão ser apresentadas mais imagens, exatamente em baixo da de referência. O objetivo principal para o utilizador vai ser achar uma relação entre a imagem de referência e as opções. No caso do Palavras a Rimar, vai ser necessário que o utilizador encontre a imagem que contém a palavra que rime com a original, enquanto no minijogo Sons Iniciais é necessário que o som inicial seja semelhante ao de referência. Um exemplo do *layout* descrito anteriormente pode ser visualizado na figura 3.7.

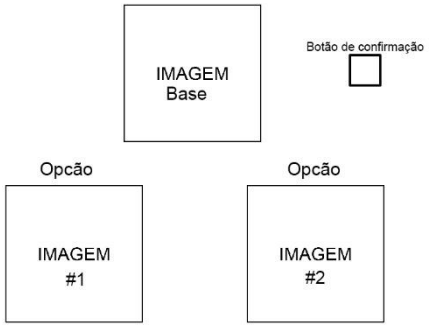


Figura 3.7 Exemplo de *Layout* do minijogo Palavras a Rimar

Quando for feita a escolha, o utilizador terá um botão para confirmar a submissão da resposta, procedendo-se depois para a validação da resposta. Em caso da resposta ser a correta será necessário dar ao utilizador um feedback positivo, caso contrário, terá de ser visualizado um feedback negativo, mas que não retire a vontade ao utilizador de continuar o minijogo. O feedback negativo foi debatido com o Centro Diferenças e acordou-se de que este não irá ter nenhum evento, apenas prossegue para o turno seguinte. Para o caso do feedback positivo foi sugerida uma personagem que reforçará positivamente a resposta. Tal e qual como os minijogos anteriores, foi decidido que estes também deverão demorar 5 turnos por cada modo de dificuldade, como está descrito na figura 3.8.

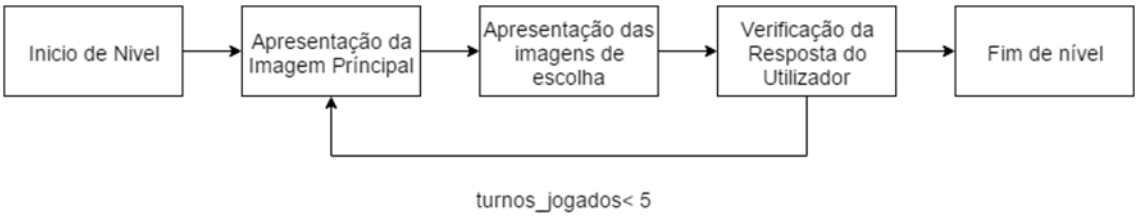


Figura 3.8 Estrutura dos minijogos Palavras a Rimar e Sons Iniciais

3.3.6 Estrutura do Minijogo Guardar os Sons

Guardar os Sons é um minijogo com a função de ensinar o utilizador a fazer associações entre um grafema e um fonema. Após o início deste, serão apresentadas 2 figuras aleatórias juntamente com um som associado, em que as figuras deverão se situar no lado esquerdo do ecrã, uma vez terminada a colocação das figuras, deverão ser posicionados dois baús à frente de cada uma, cada um representando o seu grafema e o objetivo será arrastar a figura para o baú para que contém o grafema que corresponde ao fonema inicial que ouviu, um exemplo de um possível *layout* deste minijogo pode ser observado na figura 3.9. Apenas existirá progresso no nível se todos os baús estiverem corretamente preenchidos.

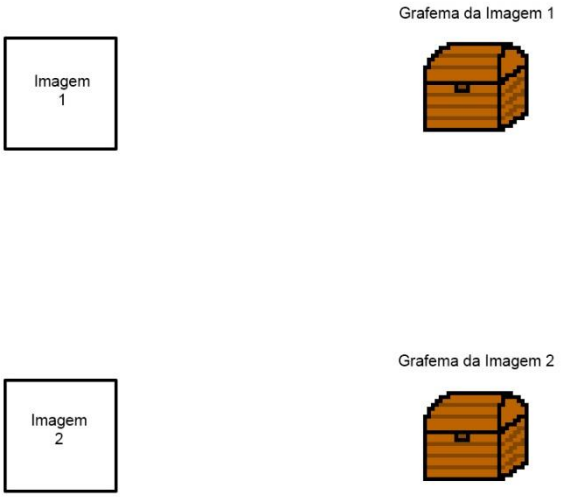


Figura 3.9 Exemplo de *Layout* do minijogo Guardar os Sons

Caso o utilizador tente colocar figuras nos baús errados, o minijogo irá tentar validar a resposta, confirmará que está errada e devolverá a imagem para a posição inicial. No caso de a resposta estar correta, o jogo irá validá-la e deverá fornecer um *feedback* positivo, como se pode verificar na figura 3.10.

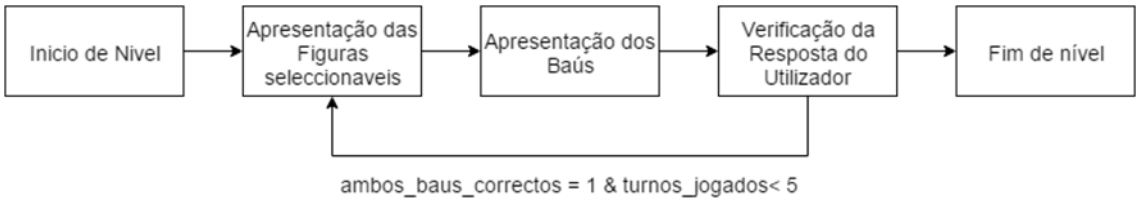


Figura 3.10 Estrutura do minijogo Guardar os Sons

3.3.7 Processo de Gravação de Dados

A gravação de dados é um processo contínuo enquanto o utilizador estiver a usar o jogo. Uma abordagem a adotar será a gravação no final de cada nível. Um sistema de pontos é útil neste caso pois permitirá detetar se o utilizador errou ou acertou num determinado minijogo. Os dados a serem guardados serão enviados em HTML através de um *POST*. Quando esta for terminada, o minijogo deverá verificar se existe mais algum nível para iniciar e, caso não haja deverá regressar para o menu principal.

O modelo da gravação de dados é descrito na figura 3.11, em que deverá ser detetado o fim de um turno e proceder ao envio dos dados. Se for verificado o fim de todos os níveis o jogo regressa para o menu principal, caso não detete carrega o nível seguinte. Se o minijogo for interrompido durante a execução, os dados são gravados, por um evento próprio para o efeito

Figura 3.11 Algoritmo de Gravação de Dados

3.3.8 API de Ligação

A API deve estar capacitada para adicionar e remover contas de utilizadores, assim como deve ser possível a gravação de dados automaticamente. Segundo os requisitos pedidos esta deverá implementar as seguintes funcionalidades:

- Capacidade de criar, eliminar e modificar utilizadores;
- Gravar Dados recebidos do jogo;
- Permitir a consulta de dados;
- Permitir *logins* e *logouts* por parte dos utilizadores.

Como foi dito no início da solução proposta, esta API vai estar responsável por todo o tipo de comunicação quer com o jogo, quer com a base de dados. O acesso ao jogo, nunca deverá ser feito de maneira direta, tal como foi demonstrado na arquitetura traçada anteriormente. O jogo nunca terá acesso direto à base de dados e nunca irá depender desta para executar, toda a interação realizada será apenas com a API.

4

Validação

Neste capítulo, serão discutidos os critérios de seleção para a implementação do modelo mencionado no capítulo anterior. Também serão explicitadas algumas das funcionalidades implementadas, assim como a análise dos dados recolhidos da implementação

4.1 Implementação

4.1.1 Escolhas Tomadas e Justificações

No decorrer do desenvolvimento desta dissertação, foram estudados diferentes métodos de como se iria proceder à implementação do modelo proposto no capítulo 3. Inicialmente pensou-se em usar uma aplicação que seria implementada no sistema operativo *Android* para executar a componente Jogo, no entanto, este método estaria a limitar o requisito de *cross-platform* (multiplataforma), ou, por outras palavras, a capacidade deste jogo estar a ser usado em computadores e dispositivos móveis. Numa abordagem seguinte, pensou-se em programar a aplicação num motor de jogo conhecido, o *Unreal Engine*, contudo este motor de jogo mostrou ser muito direcionado para jogos 3D, sendo esta solução, um jogo 2D, não faria sentido usar-se este motor de jogo. Como anteriormente foi falado, este iria também limitar o mesmo requisito que o seu predecessor. Após ter-se vetado o uso do famoso motor de jogo *Unreal Engine*, decidiu-se verificar como seria o comportamento do motor *Unity*, compilando-o para ser executado em *browser*. Inicialmente todos os testes apontaram para um grande sucesso até ao ponto em que se tentou compilar uma versão inicial, visto que anteriormente apenas se tinha produzido versões teste e todas executadas sobre o ambiente de simulação desta ferramenta. Quando o processo de compilação terminou, obteve-se um conjunto de erros que apontavam para funções que foram usadas e estavam documentadas pelo *Web site* oficial da *Unity* como funções que apenas podem ser usadas na versão PRO. Assim devido a razões monetárias, decidiu abandonar-se o uso do motor *Unity*.

Uma vez abandonada a possibilidade do uso do motor de jogo anterior, decidiu investir-se em motores de jogo especializados em HTML 5. Após a recolha dos motores de jogo existentes para HTML5 decidiu-se excluir *o Construct 2* devido à falta de capacidade de personalização, uma vez que este usa um sistema de *drag-and-drop* quer para a colocação de *sprites* no jogo, quer para a implementação de funcionalidades, sendo estas pré-feitas de maneira a reduzir a quantidade de programação da parte do criador. Outro fator que influenciou esta ferramenta pela negativa é o facto de esta ser paga.

Como foi mencionado no capítulo 2, o motor de jogo *ImpactJS* embora apresente diversas vantagens tais como um editor especializado para criação e edição de mapas ou níveis, juntamente com a deteção e personalização das definições do jogo automaticamente consoante o dispositivo.

No entanto, lamentavelmente não se encontra bem documentado e por vezes é difícil encontrar informação específica. Outra desvantagem será o fator monetário porque esta ferramenta é paga.

Em relação à ferramenta *CreateJS*, verificou-se que esta ferramenta está muito bem equipada e seria uma boa escolha para a implementação da aplicação. Contudo, após uma pesquisa extensiva, verificou-se que esta ferramente apresenta problemas de não ser exclusiva para o fabrico de jogos e possui vários motores separados. A maioria dos utilizadores usa unicamente o motor *EaselJS* visto ser este o motor responsável de dar a capacidade ao utilizador para desenhar no ecrã. Continuando com a investigação, revelou-se que esta ferramenta seria mais indicada para criadores e utilizadores de *FLASH* que desejavam uma alternativa entrando no universo HTML5. O tipo de programação foi desenhado de maneira semelhante ao *FLASH* de maneira a facilitar a transição para o uso desta ferramenta, apresentando um problema para novos utilizadores que não tenham tido um contacto anterior com este tipo de programação.

A ferramenta escolhida foi o *PhaserJs*, visto que não tem qualquer tipo de separação de motores comparando com o *CreateJS*, é *open source*, contém uma grande comunidade que fornece feedback constante de bugs encontrados e ajuda na manutenção geral da ferramenta. Outro fator a considerar foi o suporte, em que o *Phaser* demonstrou ter o melhor, em relação às ferramentas nomeadas anteriormente. Em suma, o único problema que este motor apresenta nesta fase *actual*, é a falta de um editor de texto e de um pré-visualizador próprio. Apresenta vantagens no tipo de *rendering* usado, podendo-se mudar automaticamente entre *CANVAS* e *WebGL* dependendo das necessidades do *browser* em questão.

A solução adoptada em relação comunicação da API foi recorrendo ao uso de *RESTful Web Services*. A decisão foi tomada devido ao facto destes serviços permitirem a evolução separada do jogo, como foi dito no capítulo 2, incluindo o uso de *standards* que evitam problemas de protocolos. Em termos de instalação sobre o ponto de vista do cliente, este fica completamente isento de investir em *software* extra, sendo apenas necessário a utilização do browser normal. Comparando com SOAP, este apresenta vantagens em todos os aspetos anteriormente falados, a maior vantagem que SOAP apresenta e que pudesse influenciar a decisão seria na componente de segurança. No entanto, decidiu-se que o *trade-off* seria desvantajoso para o futuro da aplicação, em suma, como foi dito anteriormente, escolheu-se serviços *RESTful* como solução a implementar na API.

4.1.2 API de Ligação

Como foi indicado no capítulo 3, a API foi implementada com o objetivo de cumprir todas as funcionalidades descritas anteriormente. A consulta de dados foi implementada na componente API para dar liberdade ao utilizador, evitando a necessidade do utilizador ligar o jogo apenas para consultar os resultados que obteve. Tal como foi justificado anteriormente, toda a comunicação com o utilizador que tenha alguma relação com dados, será toda realizada através da API. A única exceção à regra será a altura em que o utilizador está a jogar.

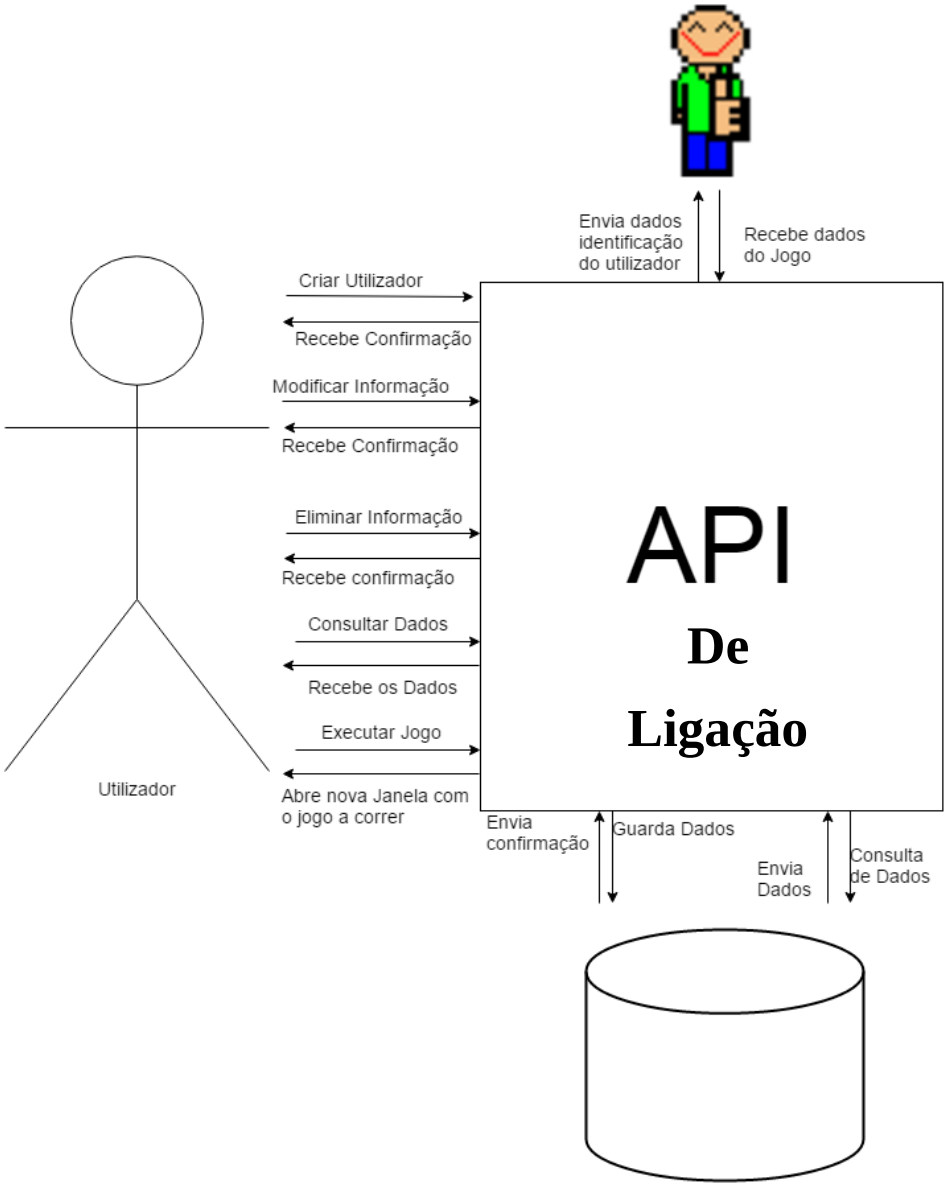


Figura 4.1 Funcionalidades da API

Como se pode verificar pela figura 4.1, quando se executa qualquer funcionalidade do lado do utilizador recebe-se feedback do lado da API. Como o jogo pode estar hospedado num local diferente do servidor, visto que este foi implementado com capacidade de escalar individualmente, quando se tenta executar o jogo a partir do servidor, vai ser aberta uma nova janela onde o jogo vai correr. A base de dados tem a função de receber, gravar e enviar dados. O processamento destes é todo feito pela API.

4.1.3 Estrutura de gravação de dados

Os dados quando são gravados seguem uma estrutura que foi definida tendo em conta as tabelas que foram criadas durante a programação da API. A tecnologia usada nesta base de dados é JSON, um formato estruturado de representação de dados. É de notar que todas as trocas de dados efetuadas durante a comunicação da API usam o formato nomeado anteriormente.

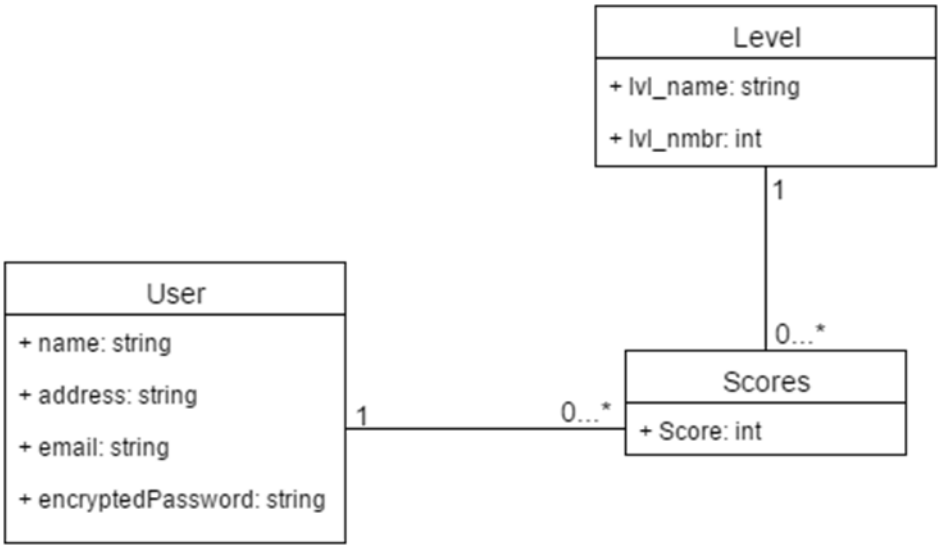


Figura 4.2 Esquema UML dos dados

Como se pode verificar na figura 4.2, estabeleceu-se uma relação 1 para N entre a tabela User e a tabela Scores, de modo a existir sempre um utilizador e garantir que o mesmo pode não ter pontuação, por não ter jogado ou pode ter N pontuações. A tabela Level foi construída, uma vez que podem existir infinitas pontuações, e os níveis não irão mudar, serão pré-estabelecidos pelo programador. Como tal, estabeleceu-se outra relação 1 para N, em que garante que existirá sempre um minijogo com um nome e um determinado número de nível como por exemplo:

Numa situação hipotética em que existe um minijogo com o nome Voar que contém 3 níveis, a tabela Level teria os seguintes parâmetros, lvl_name:Voar e o lvl_nmbr poderia variar do 1 ao 3 consoante o registo que fosse necessário. Imaginando que o utilizador fazia 20 pontos no primeiro nível, 30 pontos no segundo nível e 10 pontos no 3º nível. A tabela Level para o primeiro nível teria os parâmetros, lvl_name com o valor Voar, lvl_nmbr com o valor 1 e a tabela scores possuiria um o parâmetro score com o valor 20. Aplicando a mesma lógica para o segundo caso, os parâmetros lvl_name, lvl_nmbr da tabela Level teriam os valores, Voar e 2, respetivamente, com o valor 30 no parâmetro score da tabela Scores. O terceiro caso seguiria a mesma linha de pensamento com os valores de lvl_name, lvl_nmbr e score sendo respetivamente Voar, 20 e 10.

4.1.4 Construção do Jogo

A construção deste jogo foi iniciada com a criação da personagem principal, cujo nome é, Simão. Este personagem vive num mundo criado à sua imagem, daí o nome da aplicação "O Mundo do Simão", onde vai viver imensas aventuras que representam os diferentes mini-jogos.

O modelo visual do personagem foi feito usando uma ferramenta *open-source online* com o nome *Piskel* (Descottes & Renaudin, 2015). Numa fase inicial este personagem tinha o nome Yes Man, sendo depois renomeado para Simão.

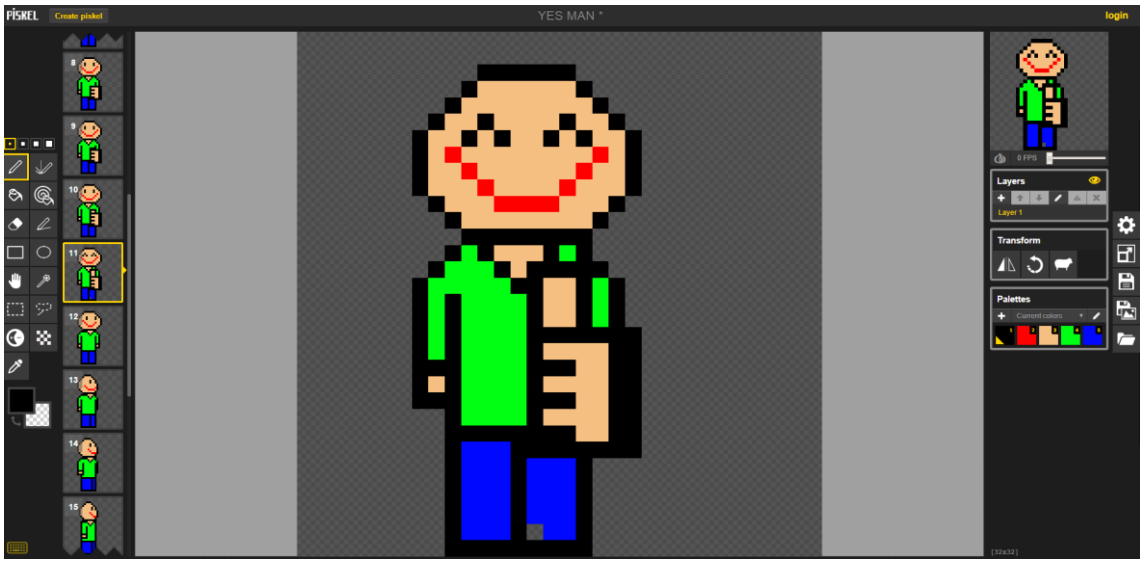


Figura 4.3 Criação da personagem Simão recorrendo ao *software Piskel* (Descottes & Renaudin, 2015)

Uma vez criado o personagem, iniciou-se o processo de criação de animações que depois iriam ser guardadas num formato e disposição que o motor de jogo conseguisse reconhecer.

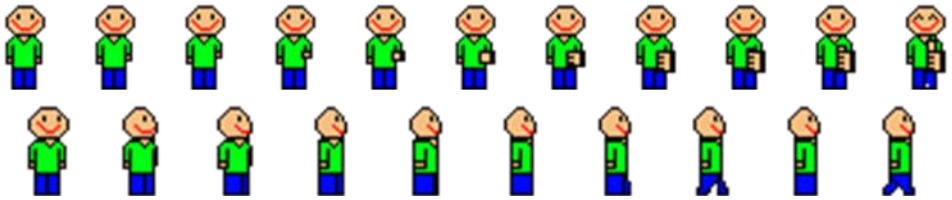


Figura 4.4 Animação andar do personagem Simão

Na figura 4.4 pode-se verificar um exemplo das animações que se foram elaboradas, podendo-se notar a quantidade de desenhos que foram necessários para fabricar uma única animação simples.

Uma vez criadas diferentes imagens e animações do personagem principal, foi então iniciada a construção da estrutura do jogo. Recorrendo-se à linguagem de programação *Javascript* e ao motor de jogo *PhaserJS*, que foi nomeado no capítulo 2 e 4.

Após diversas ponderações e reuniões no Centro Diferenças, decidiu-se que o menu principal deveria ser simples, fluído, de fácil utilização e com a informação organizada e de fácil visualização.

Figura 4.5 Menu príncipal do jogo

Como se pode observar na figura 4.5, o menu principal apresenta as 7 atividades que foram mencionadas no capítulo 3, permitindo, de modo fácil e simples, aceder a qualquer atividade. Uma vez selecionada, o jogador irá ver um pequeno vídeo de introdução onde o personagem Simão entra por uma porta para aceder a uma parte do mundo dele.

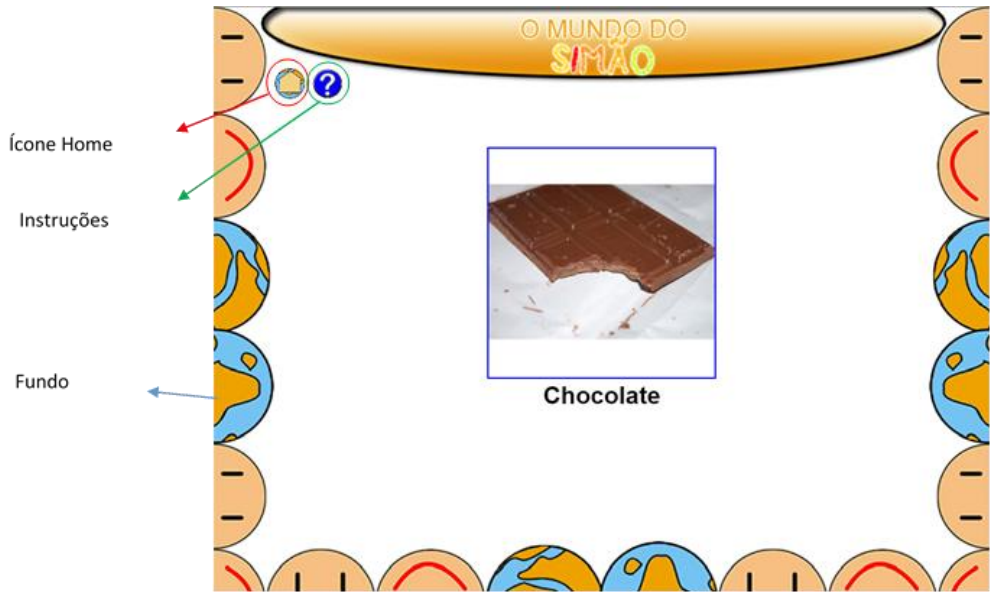


Figura 4.6 Exemplo de um minijogo e o display apresentado

Existem algumas características comuns na área de jogo que abrangem todas a atividades. Ao analisar-se a figura 4.6, é possível verificar que existem três locais destacados, o botão com o *Home* que permite ao jogador voltar ao menu principal a qualquer altura do jogo, sem perder a pontuação que tinha nesse momento. O botão Instruções que quando pressionado mostra uma janela com as instruções do jogo, podendo-se observar uma imagem desta situação no anexo 1.

Por último o fundo, que foi feito de forma a não ser extremamente distrativo mas que transmitisse a ideia de ser um jogo orientado para as crianças. Após a criação do jogo e de todos os minijogos, verificou-se que existia um pequeno atraso na colocação das imagens no ecrã. Chegou-se à conclusão que seria necessário um estado anterior ao menu principal, a fim de permitir o carregamento das imagens. Desta maneira conseguiu-se acelerar o processo de apresentação das imagens quando é iniciado o estado menu, visto que este já tem disponíveis todos os recursos que precisa.

4.1.5 Construção da API

Uma vez finalizada a construção do jogo, procedeu-se então à construção da API descrita no capítulo 3 e 4. Numa fase inicial, decidiu-se abandonar a ideia simplista de que a API teria apenas a capacidade de servir de ponte, entre o jogo e a base de dados, passando assim a servir como um meio de interação para o jogador consultar dados pessoais e dados recolhidos pelo jogo.

Foi então criada uma página HTML que disponibilizasse diversas opções ao jogador, tais como registo, login, informações sobre a aplicação e acesso ao jogo falado no capítulo anterior. É necessário o utilizador registrar-se para poder usufruir do jogo, a razão deste estar implementado desta forma foi para se conseguir acompanhar o desenvolvimento do jogador durante a utilização da ferramenta, como se pode observar na figura 4.7.

Assim que o jogador se registrar vai ser transportado para o seu perfil, onde vão ser apresentados os dados pessoais deste, juntamente com as pontuações realizadas no dia, como se encontra ilustrado na figura 4.8. As pontuações presentes no perfil sofrem um *reset* ao fim de 24 horas. No entanto, os registos permanecem na janela desempenho que pode ser acedida ao se pressionar o botão Ver resultados que está presente no perfil de cada jogador.

Outras opções presentes no perfil do jogador são a edição da informação pessoal, iniciar o jogo e *logout*. A edição de informação pessoal possibilita ao jogador alterar a sua morada e o seu nome, a opção iniciar jogo, tal como foi dito anteriormente no capítulo 4, vai abrir uma janela onde vai ser executado o jogo, e finalmente, a opção l*ogout* que permite ao jogador terminar a autenticação na aplicação.

A opção para ver todos os utilizadores apenas está presente no modo administrador, permitindo eliminar, editar e visualizar quaisquer perfis que estejam registrados na aplicação. A qualquer momento o jogador pode consultar o histórico dos seus scores desde o dia de registo, dando-lhe assim uma noção do seu desempenho e fornecendo informação importante das áreas em que melhorou e as que devem ser melhoradas, como se pode visualizar no anexo 2.

Figura 4.7 Página Inicial

Figura 4.8 Página de Perfil

4.2 Validação da comunicação API com a base de dados

A comunicação da API com a base de dados pode ser verificada ao visualizar se houve troca de informação. Tanto pela API enviar os dados corretamente ou por a base de dados estar a receber os dados todos. Para esta verificação serão enviados diversos registos para a base de dados e irão ser alterados valores de alguns parâmetros desses registros.

Posteriormente será testado o envio de dados para a API, onde se verificará se a API consegue receber e interpretar corretamente os dados recebidos e se não existe existe nenhum erro no envio destes.

Para efetuar o primeiro teste foram introduzidos os 15 registros na API com o objetivo de testar se a base de dados estaria a receber a informação corretamente.

Tabela 4.1 Tabela de Registros Introduzidos na API

Registro	**Nome**	**Morada**	***E-mail***	***Password***
1	Luis	Rua do Sol	luis@luisinho.com	123456
2	Carlos	Avenida Antiga	c.arlos@vida.pt	abcdefg
3	Paulo	Avenida da Liberdade	p.liberdade@yahoos.com	151211
4	Rodrigo	Rua do Campo	rodrig@mail.pt	potente
5	Maria	Avenida de Marte	maria@marciana.pt	regra5
6	Hélio	Rotunda da Calçada	helio@rotundacalc.com	comunica
7	Vasco	Avenida da Antena 10	vasco@naantena10.pt	transmite
8	Bob	Rua Inglesa	bob@inglesadarua.pt	england
9	Pedro	Calçada Generosa	calcada@pedrogen.com	calcada
10	Sara	Rua da Poesia	poeta@sara.pt	soupoeta
11	Nádia	Avenida Holandesa	holanda@nádia.pl	holandah
12	António	Rua Sicale	sicale@cafedoantonio.com	contactoc
13	Marilia	Avenida Agricola	campo@mariliafarm.com	agromaril
14	João	Rua Delicada	delicadeza@eojoao.com	joaodelic
15	Rúben	Avenida da Fé	rubendefe@igreja.com	rubenfei

Uma vez introduzidos todos os registos da tabela 4.1, foi-se verificar o conteúdo da base de dados. Confirmou-se que a base de dados conseguiu gravar todos os registros sem qualquer tipo de erros como se pode verificar na figura 4.9.

```
{
  "name": "Luis",
  "address": "Rua do Sol",
  "email": "luis@luisinho.com",
  "id": 1,
  "encryptedPassword": "$2a$10$VfXbbF9di485EkH26sQX/.k1Ce.XJGO5M7UXo6rXGzRiYyvMpr5VG",
  "createdAt": "2015-11-25T09:12:55.156Z",
  "updatedAt": "2016-02-25T07:11:54.849Z"
},
{
  "name": "Carlos",
  "address": "Avenida Antiga",
  "email": "c.arlos@vida.pt",
  "id": 2,
  "encryptedPassword": "$2a$10$ljtVjasCHawZXJ2xavmUSOpaEzZld/w16yeigvCfcwPYQZu27v80S",
  "createdAt": "2015-11-25T09:13:52.234Z",
  "updatedAt": "2016-02-25T07:12:31.566Z"
},
{
  "name": "Paulo",
  "address": "Avenida da Liberdade",
  "email": "p.liberdade@yahoos.com",
  "id": 3,
  "encryptedPassword": "$2a$10$NH/LIN4pJ/PkX7KzSrdbd./JZ3G7Q101KX7y7p/EbIaXSxIzULpsa",
  "createdAt": "2015-11-25T09:14:34.557Z",
  "updatedAt": "2016-02-25T07:12:58.790Z"
},
{
  "name": "Rodrigo",
  "address": "Rua do Campo",
  "email": "rodrig@mail.pt",
  "id": 4,
  "encryptedPassword": "$2a$10$IAtNhjoKHwkq75cmda/./OEQuWUZfL.MHob57.C1HsZpnC20ouRl6",
  "createdAt": "2015-11-25T09:16:34.145Z",
  "updatedAt": "2016-02-25T07:13:22.532Z"
},
{
  "name": "Maria",
  "address": "Avenida de Marte",
  "email": "maria@marciana.pt",
  "id": 5,
  "encryptedPassword": "$2a$10$anMkff.HiEh8x/HovcOolehTwshhd6rcVwq0MwF3WTAjGuziYEMpO",
  "createdAt": "2015-11-25T09:19:02.469Z",
  "updatedAt": "2016-02-25T07:13:50.145Z"
},
```


Figura 4.9 Conteúdo na base de dados

Como teste final relativo à comunicação da API com a base de dados, irá proceder-se à recolha da informação que está guardada para ser processada e apresentada no ecrã pela API. Se existir algum erro pode ser por a base de dados estar a enviar informação errada, ou por existir algum erro no processamento de dados pelo qual a API é responsável.

Figura 4.10 Resultado da recolha de informação

Os resultados parciais da recolha dos dados da base de dados encontram-se na figura 4.10, a figura completa esta está presente no anexo 3. Como se pode verificar na figura 4.10 e no anexo 3, todos os dados apresentados correspondem aos presentes na tabela 4.1. O campo *password* não é mostrado na lista de utilizadores fornecida pela API, no entanto, todos os utilizadores foram testados através da execução do login de cada um utilizando os dados presentes na tabela 4.1. Obtiveram-se resultados positivos do teste de login para cada utilizador e verificou-se que todas as passwords correspondiam corretamente a cada utilizador segundo a tabela 4.1. Pode-se concluir que o teste foi um sucesso e que não existem erros de receção ou envio de dados na comunicação da API com a base de dados.

4.3 Validação da comunicação da API com o Jogo

De forma a garantir que a comunicação entre a API e o jogo está a funcionar como foi pensada, devem ser feitos diversos testes de envio de informação da API para o jogo e vice-versa.

Para se realizar este teste, decidiu-se realizar um login na API e em seguida executar o jogo. Numa fase inicial irá ser verificado se o jogo consegue receber a informação correta do utilizador de modo a o conseguir identificar.

Figura 4.11 Login usado para o teste

Figura 4.12 Resultado obtido na componente do jogo

O registo usado para este teste foi o login do utilizador rodrigo que corresponde ao registo número 4 da tabela 4.1. pode verificar na figura 4.11, o jogo foi executado e na figura 4.12 é possível ver que o jogador foi corretamente identificado, provando assim que a receção de dados por parte da API está a funcionar sem quaisquer problemas.

Tendo em conta que o jogo consegue receber dados da API com sucesso, fica apenas a faltar, a verificação do envio de dados por parte do jogo para a API. Como teste, atribuíram-se pontuações em minijogos diferentes que vão ser enviadas para a API de modo a verificar se existe algum erro no envio de dados do jogo.

Tabela 4.2 Pontuações a serem enviadas

MiniJogos	Pontuação
Palavras a Rimar	30
Contar os Sons	10
Contar os Bocadinhos	20

Assim que foram enviadas as pontuações que estão apresentadas na tabela 4.2, acedeu-se ao perfil do jogador com o objetivo de se verificar se as pontuações obtidas no jogo foram transmitidas com sucesso para a API.

Figura 4.13 Perfil do Utilizador após envio de dados do jogo

Como se pode verificar na figura 4.13, todas as pontuações presentes na tabela 4.2 estão presentes nos registos do jogador, podendo-se concluir que a comunicação da API com o jogo está a funcionar corretamente sem ter apresentado quaisquer tipos de erros durante a execução dos testes.

4.4 Validação Online

Uma vez realizados todos os testes localmente, decidiu-se que seria altura de testar o comportamento do sistema *online*, de maneira a provar que o funcionamento se mantinha igual ao observado localmente.

Para realizar este teste, recorreu-se ao uso da plataforma *online* Heroku (Heroku, 2016) usada para hospedar sites diversos, como se pode verificar na figura 4.14.

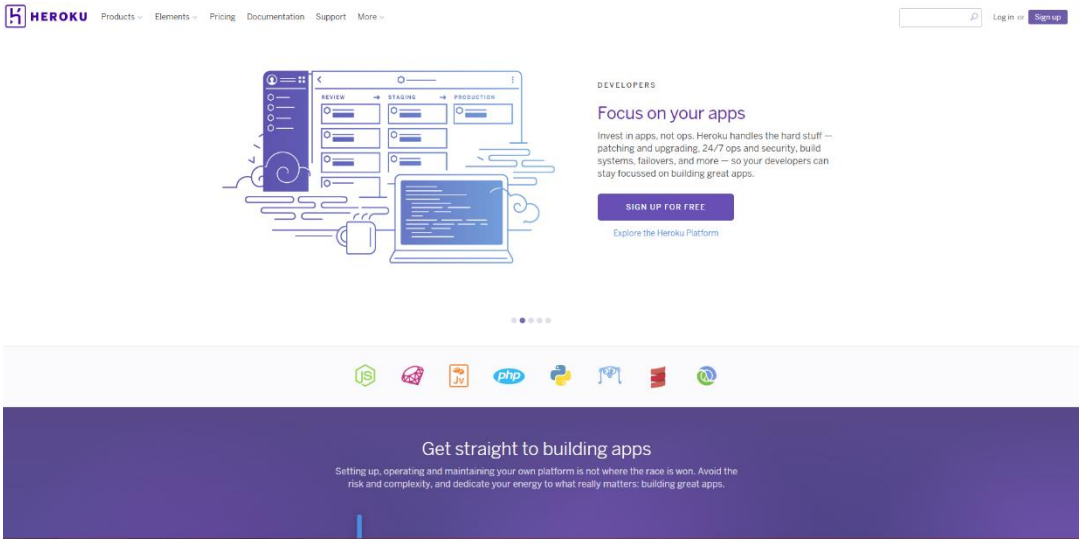


Figura 4.14 Plataforma online Heroku usada para hospedar o projecto Mundo do Simão

Tal como esperado, o projeto foi hospedado com sucesso, apresentando o mesmo comportamento durante os testes efetuados localmente. O *site* esteve disponível durante 48 horas (do 22 de Março ao 24 de Março) com a hiperligação: https://mundodosimao.herokuapp.com/ Esta foi acedida por computadores e diversos dispositivos móveis, entre eles:

- Telemóvel de 5 polegadas, Huawei P8 Lite com o sistema operativo Android 5.0.1.
- *Tablet* de 10 polegadas, Asus Zenpad com o sistema operativo Android 5.1.
- *Tablet* de 10 polegadas, Asus Tranformer Pad (TF-300T) com o sistema operativo Android 4.2.1.
- *Tablet* de 7 polegadas, Asus Nexus 7 com o sistema operativo Android 5.1.
- Telemóvel de 5 polegadas, Huawei G700 com o sistema operativo Android 4.2.1.
- Ipod Touch de quarta geração com 3.5 polegadas com o sistema operativo IOS.

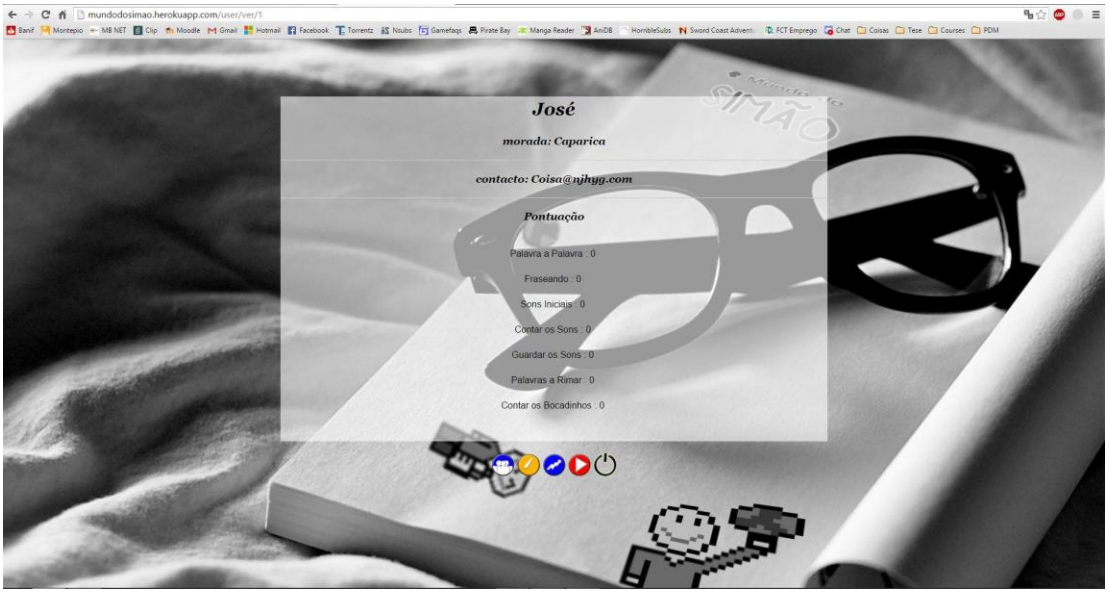


Figura 4.15 Perfil de um utilizador no projeto Mundo do Simão acedido por computador.

Como se pode verificar pela Figura 4.15, o perfil apresenta a mesma disposição e comportamento esperado dos testes efetuados localmente. Contudo, o jogo também terá de ser acedido para se garantir que todas as componentes estão a funcionar corretamente.

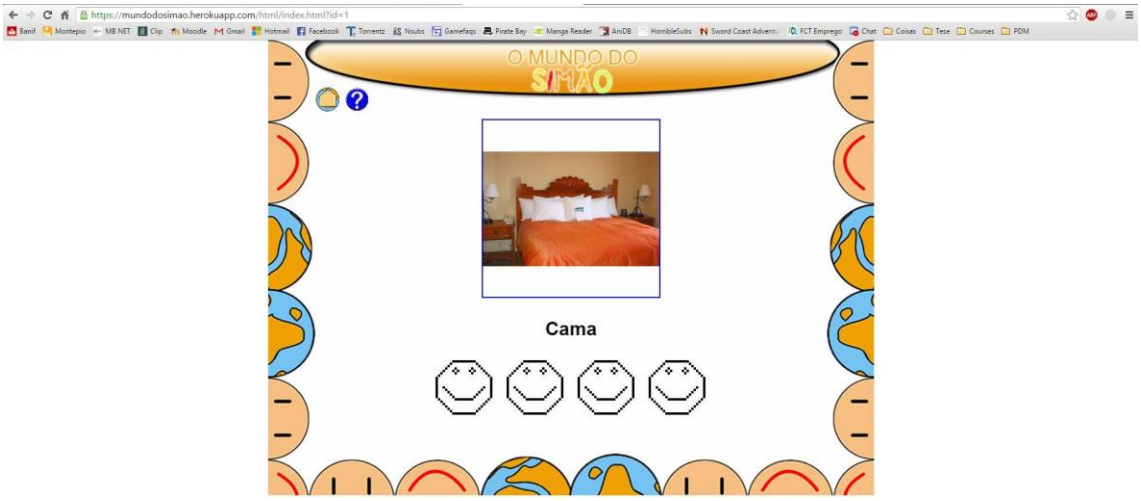


Figura 4.16 Jogo no projeto Mundo do Simão acedido por computador

Embora os testes no computador tenham correspondido às expectativas, falta testar os dispositivos móveis, sendo este um objetivo importante para a validação do funcionamento do sistema *online*. Iniciou-se os testes com os dispositivos mais pequenos, neste caso os telemóveis, pois apresentam menor dimensão de ecrã.

Figura 4.17 Imagens do perfil no projeto Mundo do Simão visualizado no Huawei P8 Lite

A sucessão de imagens presentes na Figura 4.17 mostram a visualização do perfil do utilizador José no telemóvel Huawei P8 Lite nomeado anteriormente, a imagem presente no topo desta sucessão de imagens mostra o ecrã que é mostrado uma vez feito o login e as imagens consecutivas mostram a evolução do ecrã à medida que se vai fazendo *scroll* deste até ao fim da página. Uma vez finalizado o acesso ao perfil, decidiu-se proceder à execução do jogo para testar o seu funcionamento num dispositivo móvel.

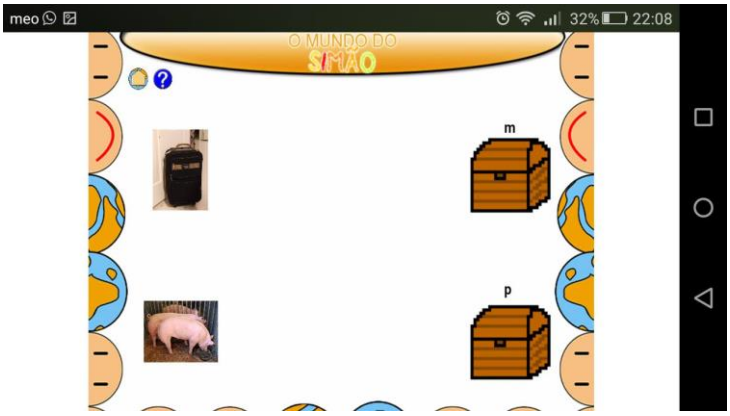


Figura 4.18 Imagem do jogo Guardar os Sons no projecto Mundo do Simão visualizado no Huawei P8 Lite

Com estes testes, concluiu-se que a aplicação está a funcionar corretamente e apresentou comportamento espetável sem qualquer tipo de surpresas, em dispositivos móveis e computadores. Um dos testes efetuados no *tablet* ZenPad encontra-se nos anexos 4 e 5, no qual consta uma figura do perfil do utilizador e uma figura do jogo Contar os Sons.

4.5 Validação dos dados experimentais

Durante o intervalo de tempo entre 5 de Outubro de 2015 e 2 de Dezembro do mesmo ano, foram recolhidos dados de 3 crianças do centro diferenças que deram uso ao projeto desta dissertação. Os dados recolhidos representam um total de 3 sessões de cada criança sendo o espaçamento entre sessões variável.

Para a análise dos dados, foram consideradas 9 sessões que continham pontuações dos minijogos, Palavra a Palavra, Fraseando, Guardar os Sons, Contar os Sons, Palavras a Rimar e Contar os Bocadinhos. O único minijogo que não possuí qualquer tipo de dados é o Sons Iniciais.

Os dados serão apresentados em tabelas, onde será indicado o número da sessão juntamente com as pontuações dependendo do minijogo. Não será nomeado qualquer nome que possa identificar o jogador em causa, sendo estes identificados da seguinte maneira, criança X, criança Y e criança Z.

Tal como foi mencionado no capítulo 3, os minijogos são constituídos por 5 turnos. Cada resposta correta vale 10 pontos, cada resposta errada vale 0 pontos, sendo que o valor máximo que uma criança pode tirar num determinado nível de um minijogo é 50 que corresponde à criança ter respondido a tudo corretamente e o mínimo é 0 que corresponde à criança ter errado todos os turnos.

Tabela 4.3 Criança X

Minijogos\DIAS	**Dia 1**	**Dia 2**	**Dia 3**	**Análise**
Palavra a Palavra	50	50	50	Manteve
Fraseando	0	0	10	Melhorou
Guardar os Sons	40	40	**x**	Manteve
Contar os Sons	50	10	**x**	Piorou
Palavras a Rimar	10	20	**x**	Melhorou
Contar os Bocadinhos	50	50	50	Manteve

O **x** que está presente na tabela 4.3 no terceiro dia, nos minijogos Guardar os Sons, Contar os Sons e Palavras a Rimar significa que não foram registados dados nesses minijogos nesse dia. A análise é feita entre sessões, tendo por exemplo a tabela 4.3. Assim que se procedeu à análise da tabela, conseguiu-se concluir que esta criança não obteve qualquer tipo de resultado nos minijogos, Guardar os Sons, Contar os Sons e Palavras a Rimar no terceiro dia. Esta criança no entanto melhorou em 2 minijogos, manteve a pontuação máxima no jogo Palavra a Palavra, Guardar Sons e Contar os Bocadinhos, piorando apenas no minijogo Contar os Sons.

Tabela 4.4 Criança Y

Minijogos\DIAS	Dia 1	Dia 2	Dia 3	Análise
Palavra a Palavra	10	10	50	Melhorou
Fraseando	0	10	10	Melhorou
Guardar os Sons	0	10	**x**	Melhorou
Contar os Sons	**x**	**x**	**x**	SDS
Palavras a Rimar	0	10	0	Piorou
Contar os Bocadinhos	0	40	30	Melhorou

Uma vez analisada a tabela 4.4, podemos concluir que esta criança não obteve qualquer tipo de resultado no minijogo Contar os Sons e Guardar os Sons no terceiro dia. Esta criança no entanto melhorou em 4 minijogos, subiu para a pontuação máxima no jogo Palavra a Palavra. Contudo, denota-se um decréscimo de pontuação no jogo Palavras a Rimar. É de notar que, embora no jogo Contar os Bocadinhos, a criança Y teve um decréscimo de pontuação no terceiro dia, o balanço final entre pontuação ganha e pontuação perdida, acaba por ser positivo (40 pontos ganhos – 10 pontos perdidos = 30 pontos ganhos).

Tabela 4.5 Criança Z

Minijogos\DIAS	Dia 1	Dia 2	Dia 3	Análise
Palavra a Palavra	0	50	x	Melhorou
Fraseando	0	10	x	Melhorou
Guardar os Sons	40	x	**x**	SDS
Contar os Sons	x	x	**x**	SDS
Palavras a Rimar	0	10	**x**	Melhorou
Contar os Bocadinhos	0	50	x	Melhorou

Por fim, a análise da Tabela 4.5 demonstra que a criança Z no dia 3 não efetuou qualquer tipo de minijogo visto que não existe qualquer tipo de pontuação nesse mesmo dia. O balanço dos 2 primeiros dias aparenta ser bastante positivo visto que, a criança passou de 0 pontos para a pontuação máxima em apenas uma sessão para o minijogo Palavra a Palavra e o minijogo Contar os Bocadinhos. Verificou-se uma melhoria menos acentuada nos minijogos Fraseando e Palavras a Rimar, verificando-se uma falta de dados no minijogo Contar os Sons e Guardar os Sons. De um modo geral, esta criança revelou que melhorou a sua pontuação em todos os jogos que participou em ambos os dias, demonstrando um grande sucesso no desempenho da aplicação.

É de notar, que quando o valor SDS (Sem Dados Suficientes) aperece em algumas análises da tabela significa que não existem dados suficientes para realizar-se uma avaliação/análise.

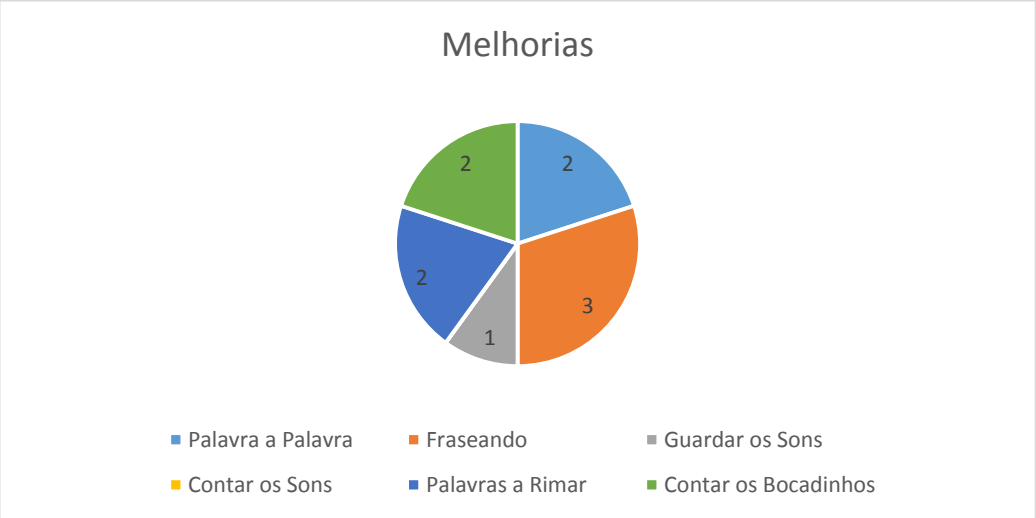


Figura 4.19 Gráfico de Melhorias em todos os minijogos

Com o gráfico da figura 4.19, podem-se observar a quantidade de melhorias presentes em todos os minijogos. A razão da cor amarela não ser visível na figura é devido ao fato desse minijogo apenas possuir avaliação negativa, com 0 melhorias. Com esta análise também se consegue concluir os minijogos Contar os Sons e Guardar os Sons devem ser melhorados, o Contar os Sons por ter obtido uma nota 100% negativa, apresentado nem melhorias nem preservação de pontos, existindo uma descida de pontuação para todos os participantes. O jogo Guardar os Sons deve ser melhorado de maneira a provocar um aumento de melhorias de pontuação.

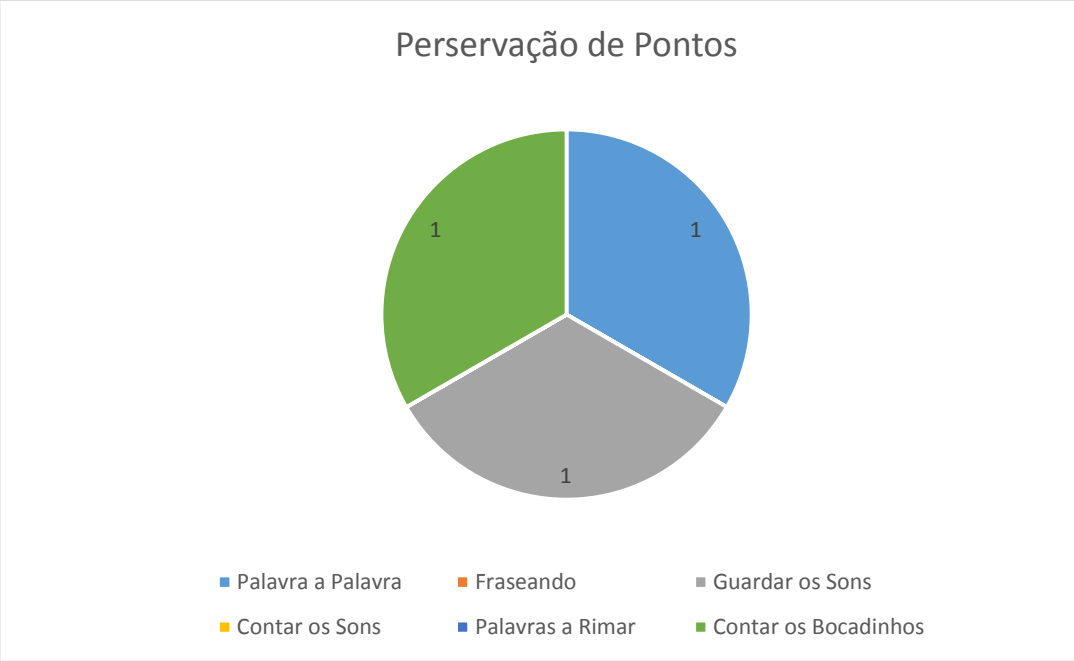


Figura 4.20 Gráfico que apresenta a quantidade de minijogos que mantiveram as pontuações

Com o gráfico da figura 4.20, pode-se concluir que em apenas 3 minijogos é que existiu a preservação das pontuações durante as sessões.

É de notar que em 2 dos minijogos, Contar os Bocadinhos e Palavra a Palavra, a nota foi mantida porque a criança em causa tinha atingido a pontuação máxima e continuou a faze-lo durante todas as sessões restantes.

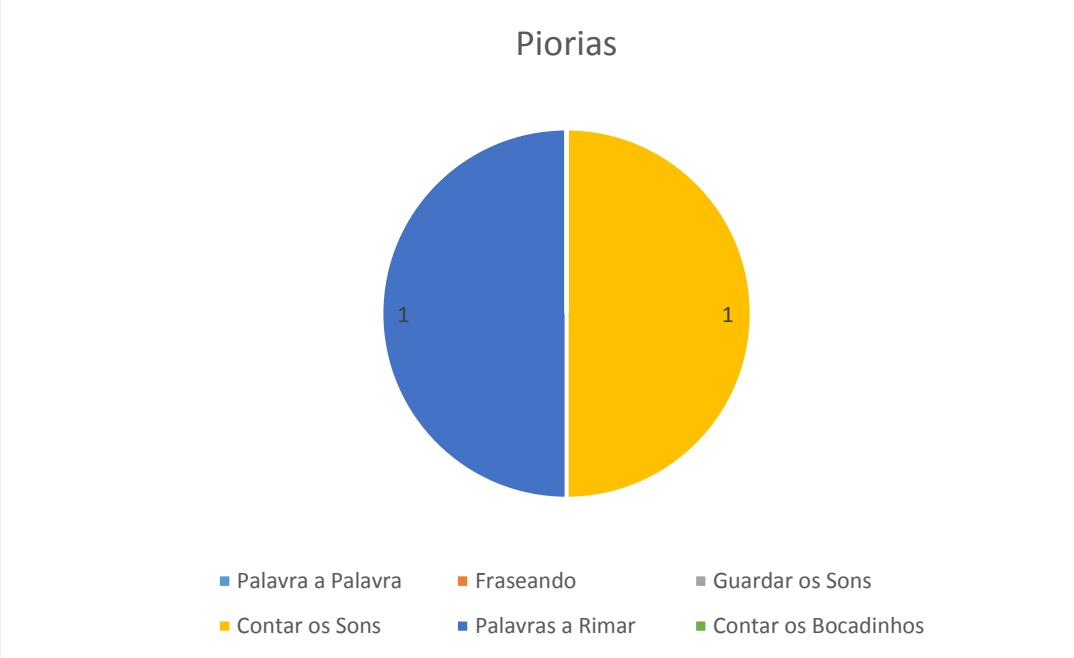


Figura 4.21 Gráfico de Piorias nos minijogos

Por fim, a figura 4.21 representa o gráfico com os minijogos em que existiram piorias de pontuação juntamente respetiva contagem de jogadores. Como era de esperar, pode-se facilmente concluir que apenas existiram 2 minijogos onde se verificaram pontuações a piorar. Os minijogos em causa são o Contar os Sons e o Palavras a Rimar. Estes jogos devem ser revistos com o objetivo de minimizar a redução de pontuações e encontrar uma alternativa que provoque o aumento das pontuações.

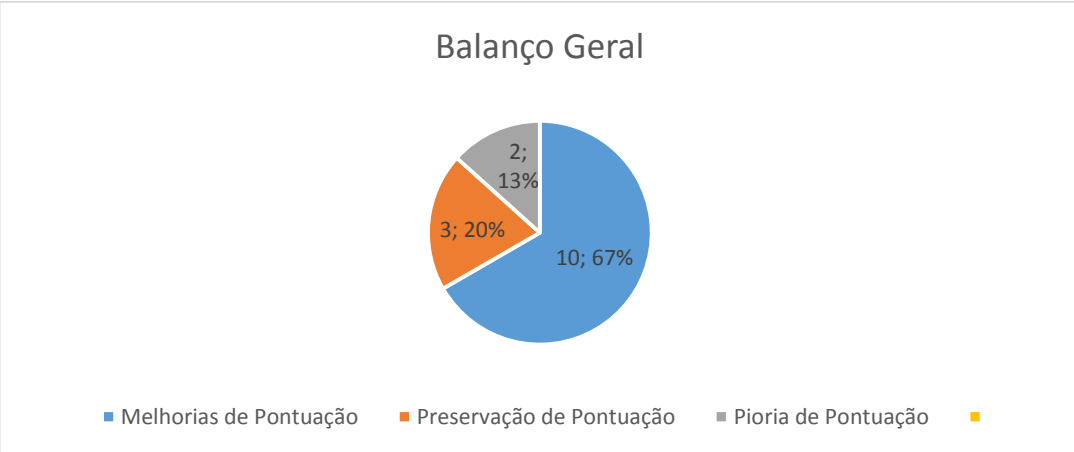


Figura 4.22 Gráfico com o balanço geral da análise de todos os minijogos

Como se pode observar pela figura 4.22, num universo de 15 amostras, o balanço geral foi positivo, existindo uma melhoria de 67% nas pontuações correspondendo a dez das quinze amostras, com uma preservação de pontuações em 20% correspondendo a três das quinze amostras e 13% de pioria na pontuação, correspondendo a 2 das quinze amostras.

Respondendo à questão colocada no capítulo 1.1, verificou-se segundo os dados experimentais obtidos que quando aplicadas as ferramentas certas no ensino para a promoção da linguagem pode-se realmente aumentar o desempenho de pessoas com problemas no neurodesenvolvimento. Bons exemplos da afirmação anterior são os mini-jogos Palavra-a-Palavra e Contar os Bocadinhos.

5

Conclusão

Neste capítulo serão abordados alguns aspetos, dificuldades e desafios levantados aquando do desenvolvimento desta dissertação, assim como um conjunto de considerações finais relacionadas com os resultados obtidos.

Em relação ao desenvolvimento de trabalhos futuros, serão apresentadas algumas sugestões de projetos e funcionalidades que poderão acrescentar valor e aumentar a eficácia e eficiência da aprendizagem adquirida com o uso do projeto desenvolvido.

5.1 Conclusões

Com esta dissertação pôde-se entender os diversos desafios que os programadores passam quando existe uma cooperação com outras instituições durante o desenvolvimento de um projeto. Assim sendo, a experiência adquirida durante esta conceção é uma mais-valia para qualquer engenheiro, tanto para o desenvolvimento futuro de um projeto, como para a experiência em cooperar com pessoas que não detêm a mesma área de especialização. Dentro dos desafios ultrapassados, um que mostrou mais relevo esteve relacionado com cooperação entre a Universidade e o Centro Diferenças, provando-se mais desafiante, tendo em conta que foi o primeiro impacto com profissionais de outras áreas e as suas perspetivas perante o projeto.

O projeto Mundo do Simão finalizou com uma nota positiva, suportado pelos resultados experimentais adquiridos, podendo assim afirmar-se que será uma nova ferramenta suficientemente poderosa a acrescentar ao "armamento" da educação de crianças com síndrome de *Down*.

Os objetivos traçados foram alcançados com sucesso. No entanto, existiu alguma dificuldade na implementação da gravação de dados devido a uns ajustes não planeados. Estes demonstraram ser necessários para que os objetivos definidos pudessem ser cumpridos.

Outro desafio que se provou um bocado mais moroso do que seria esperado inicialmente foi o *design* da interface e os respetivos ajustes perante os objetivos pré-definidos anteriormente. Acabou por se concluir que por vezes, a pessoa que está responsável pela componente de *design* não tem noção das limitações que o utilizador final pode possuir, tornando-se assim essencial uma constante comunicação e cooperação com as instituições especializadas, neste caso o Centro Diferenças.

Fazendo um balanço final, esta dissertação foi realizada com sucesso, cumprindo-se os objetivos propostos e provou o seu sucesso com os resultados experimentais que foram obtidos. A resposta obtida à questão do capítulo 1 foi conseguida, embora tenham existido resultados experimentais positivos, deve-se ter cuidado com a implementação das mecânicas de jogo, visto que, no caso de serem mal implementadas podem prejudicar o progresso do jogador, como foi observado no jogo Palavras a Rimar, contudo, ficou provado com os dados obtidos que as

mecânicas de jogo conseguem aumentar o desempenho da aprendizagem em pessoas com problemas no neurodesenvolvimento.

5.2 Contribuições

O autor gostaria de agradecer e de sublinhar o contributo da instituição "Centro Diferenças" por me ter dado a possibilidade de desenvolver este projeto, assim como todo o tempo dispensado para a discussão de ideias e mecânicas a implementar durante a fase de desenvolvimento desta dissertação.

5.3 Trabalhos Futuros

Após a análise prática de ambos os resultados e o centro diferenças, chegou-se à conclusão que existiriam pontos que poderiam ser melhorados neste projeto.

Um melhoramento apontado seria um sistema ou um algoritmo de deteção de voz que teria como função complementar uma mecânica já implementada em diversos jogos. Este melhoramento torna o projeto mais automatizado e permite um despiste de alguns falsos positivos que possam ocorrer.

O acréscimo de um possível criador de níveis personalizados seria uma boa adição a este projeto, visto que, daria a capacidade ao administrador de personalizar programas específicos para cada jogador.

Uma das problemáticas que esteve sempre no centro deste projeto foi o fator de cativar o jogador, que neste caso é uma criança, para jogar. Tornou-se óbvio que embora este projeto tenha uma grande interatividade verificou-se que a necessidade de uma quantidade superior desta, de modo a aumentar a motivação do jogador em cumprir objectivos. Uma possível solução passará por uma revisão do *design* do personagem principal, assim como a história por detrás do mesmo.

A imersão tal como foi dito no capítulo 2, tanto é uma dádiva como uma perdição, isto porque pode cativar e desmotivar o jogador dependendo da sua implementação. Este projeto focou-se bastante neste aspeto, contudo, uma melhoria neste campo poderá aumentar o desempenho na aprendizagem do jogador e a sua motivação.

Por fim, uma funcionalidade que traria mais-valia a este projeto passaria por implementar uma interface que permita o acréscimo de sons e imagens de maneira fácil para o administrador final.

6 Bibliografia

Adams, E. (2010). *Fundamentals of Game Design 2nd Edition.* New Riders.

Apatris21. (2015 de Novembro de 23). *ProSensi Acções de Sensibilização Informação*. Obtido de Apatris21: http://www.apatris21.org/

Armstrong, L. (2000). Distance Learning: An Academic Leader's Perspective on a Disruptive Product. *Change: The Magazine of Higher Learning*, (pp. 20-27).

BBC. (15 de Janeiro de 2016). *BBC News*. Obtido de BBC News: http://news.bbc.co.uk/2/hi/technology/6572711.stm

Bunchball. (2010). *Gamification 101: An Introduction to the Use of Game Dynamics to Influence Behavior.* BunchBall.

Burgoyne, K. (2009). *Reading interventions for children with Down syndrome.* Down Syndrome Research and Practice.

Burke, B. (2014). *Gamify How Gamification Motivates People to do Extraordinary Things.* Gartner, Inc.

Cerami, E. (2002). *Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL.* O'Reilly Media.

Connolly, T., & Stansfield, M. (2006). Using Games-Based eLearning Technologies in Overcoming Difficulties in Teaching Information Systems. *Journal of Information Technology Education*, (pp. 459-476). Scotland, Inglaterra.

Crawford, C. (1984). *The Art of Computer Game Design.* McGraw-Hill/Glencoe.

Cunha, M. I., & Santos, L. M. (2007). *Aprendizagem Cooperativa na Deficiência Mental (Trissomia 21).* Porto: Revista Cadernos de Estudo Nº5.

Davey, R. (2 de Setembro de 2015). INTERVIEW TO RICHARD DAVEY, CREATOR OF PHASER. (Gamepix, Entrevistador)

Descottes, J., & Renaudin, V. (12 de Fevereiro de 2015). *Piskel | Free online Sprite Editor*. Obtido de Piskel: http://www.piskelapp.com/

Diferenças. (11 de Fevereiro de 2016). *Diferenças | Centro de Desenvolvimento Infantil*. Obtido de Diferenças: http://diferencas.net/

Doglio, F. (2015). *Pro REST API Development with Node.js.* Uruguai: Apress.

Enders, B. (2013). *Gamification, Games, and Learning: What Managers and Practitioners Need to Know.* The eLearning Guild.

Epignosis LLC. (2014). *E-Learning Concepts, Trends, Applications.* São Francisco, Califórnia, CA 94104 Estados Unidos da América: Epignosis LLC.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Universidade da Califórnia, Irvine.

Frasca, G. (2001). *Videogames of the Opressed: Videogames as a means for critical thinking and debate.* Geórgia: Georgia Institute of Technology.

Gomes, A. (6 de Novembro de 2015). aula de Tecnologias de Jogos de Vídeo. Universidade da Beira Interior, Covilhã, Portugal.

GooTechnologies. (20 de Novembro de 2015). *Goo Create - Goo's 2013 State of Browser Gaming Results Revealed*. Obtido de Goo Create: http://goocreate.com/goos-2013-state-of-browser-gaming-results-revealed/

Granic, I., Lobel, A., & Engels, R. C. (2014). The Benefits of Playing Video Games. *American Psychologist* (pp. 66-78). American Psychological Association.

Heroku. (12 de Fevereiro de 2016). *Could Application Platform| Heroku*. Obtido de Heroku: https://www.heroku.com/

Horachek, D. (2014). *Creating E-Learning Games With Unity*. Birmingham B3 2PB, UK: Packt Publishing.

Instituto Nacional de Estatística. (Janeiro de 2015). *www.ine.pt.* Obtido de Instituto Nacional de Estatística: https://www.ine.pt/ngt_server/attachfileu.jsp?look_parentBoui=249615582&att_display=n&att_download=y

Kent, S. L. (2001). *The Ultimate History of Video Games: From Pong to Pokémon and Beyond - The Story Behind the Craze that Touched Our Lives and Changed the World.* Nova Iorque: Three Rivers Press.

Kose, U., & Arslan, A. (2015). E-Learning Experience With Artificial Intelligence Supported Software: An International Application on English Language Courses. *GLOKALde*, (pp. 61-75). Turquia.

Koster, R. (2006). *A Theory of Fun for Game Design.* Paraglyph Press.

Makesys. (24 de Novembro de 2015). *Makesys Fábrica de Software*. Obtido de Makesys Fábrica de Software: http://www.makesys.com.br/

Manderscheid, B. (2014). *Beginning HTML5 Games with CreateJS.* Apress.

Marczewski, A. (2013). *Gamification: A Simple Introduction.* Andrzej Marczewski.

Massé, M. (2012). *REST API Design Rulebook.* O'Reilly Media.

Michael, D., & Chen, S. (2006). *Serious Games: Games that Educate.* Canadá: Course Technology.

Microsoft. (9 de Novembro de 2015). *Xbox One | Site Oficial*. Obtido de Xbox One: http://www.xbox.com/pt-PT/xbox-one

Moreira, L. M., & Gusmão, F. A. (2002). Aspectos genéticos e sociais da sexualidade em pessoas com síndrome de Down. *Revista Brasileira de Psiquiatria*, (pp. 94-99).

Nintendo. (9 de Novembro de 2015). *Wii U from Nintendo - Official Site - HD Video Game Console*. Obtido de Nintendo Wii: http://www.nintendo.com/wiiu

Node.js. (21 de Novembro de 2015). *Node.js*. Obtido de Node.js: https://nodejs.org/en/

O'Donoghue, J., Singh, G., & Green, C. (2004). A comparison of the advantages and disadvantages of IT based education and the implications upon students. *Interactive Educational Multimedia*, (pp. 63-76).

Olson, C. K. (2010). Children's Motivations for Video Game Play in the Context of Normal Development. *Review of General Psychology* (pp. 180-187). American Psychological Association.

Oracle. (2013). *The Java EE 6 Tutorial.* Oracle.

Oracle. (2 de Novembro de 2015). *What Are RESTful Web Services? - The Java EE 6 Tutorial*. Obtido de The Java EE 6 Tutorial: http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

Pais21. (15 de Janeiro de 2016). *Pais 21 - Grupo de Pais e Amigos*. Obtido de Pais 21: http://pais21.pt/pt/

Pamplona, V. F. (22 de Outubro de 2015). *javafree.uol.com.br*. Obtido de Javafree: http://javafree.uol.com.br/artigo/871485/Web-Services-Construindo-disponibilizando-e-acessando-Web-Services-via-J2SE-e-J2ME.html

Paul, P. S., Goon, S., & Bhattacharya, A. (2012). History and Comparative Study of Modern Game Engines. *International Journal of Advanced Computer and Mathematical Sciences*, (pp. 245-249).

Prensky, M. (2007). *Digital Game-Based Learning.* St.Paul, Minnesota: Paragon House.

Rabin, S. (2010). *Introduction to Game Development.* Boston, Estados Unidos de América: Course Technology.

Rollings, A., & Morris, D. (2004). *Game Architecture and Design.* New Riders.

Rutten, L. (25 de Fevereiro de 2014). HTML5 game company founder: we don't need to innovate, we just need to be good. (T. i. Asia, Entrevistador)

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology* (pp. 54-67). Academic Press.

Salen, K., & Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals.* Massachussets Institute of Technology.

Scirra. (21 de Novembro de 2015). *Construct 2*. Obtido de Construct 2: http://www.scirra.com/construct2

Sheehy, P. (20 de Outubro de 2015). *WowinSchool*. Obtido de World of Warcraft in School: http://wowinschool.pbworks.com/w/page/5268731/FrontPage

Skinner, G., Mcnie, L., & Gorgichuk, W. (22 de Novembro de 2015). *CreateJS| A suite of Javascript libraries and tools designed for working with HTML5*. Obtido de CreateJS: http://www.createjs.com/

Snell, J., Tidwell, D., & Kulchenko, P. (2001). *Programming Web Services with SOAP.* O'Reilly Media.

Sony. (10 de Novembro de 2015). *PS4| Sistema da próxima geração da Sony| Playstation*. Obtido de Ps4: https://www.playstation.com/pt-pt/explore/ps4/

Sousa, F. P. (2015). Criação de framework REST/HATEOAS Open Source para desenvolvimento de APIs em Node.js. (pp. 36-37). Porto: Faculdade de Engenharia da Universidade do Porto.

Spuy, R. v. (2015). *Learn Pixi.js Create Great Interactive Graphics for Games and the Web.* Apress.

Stowe, M. (2015). *Undisturbed REST.* MuleSoft.

Subagio, A. (2014). *Learning Construct 2 - Design and create your own engaging, extensible, and addictive game using Construct 2.* Packt Publishing.

Szablewski, D. (1 de Maio de 2011). Impact Game Engine. *Game Developer Magazine*, pp. 31-32.

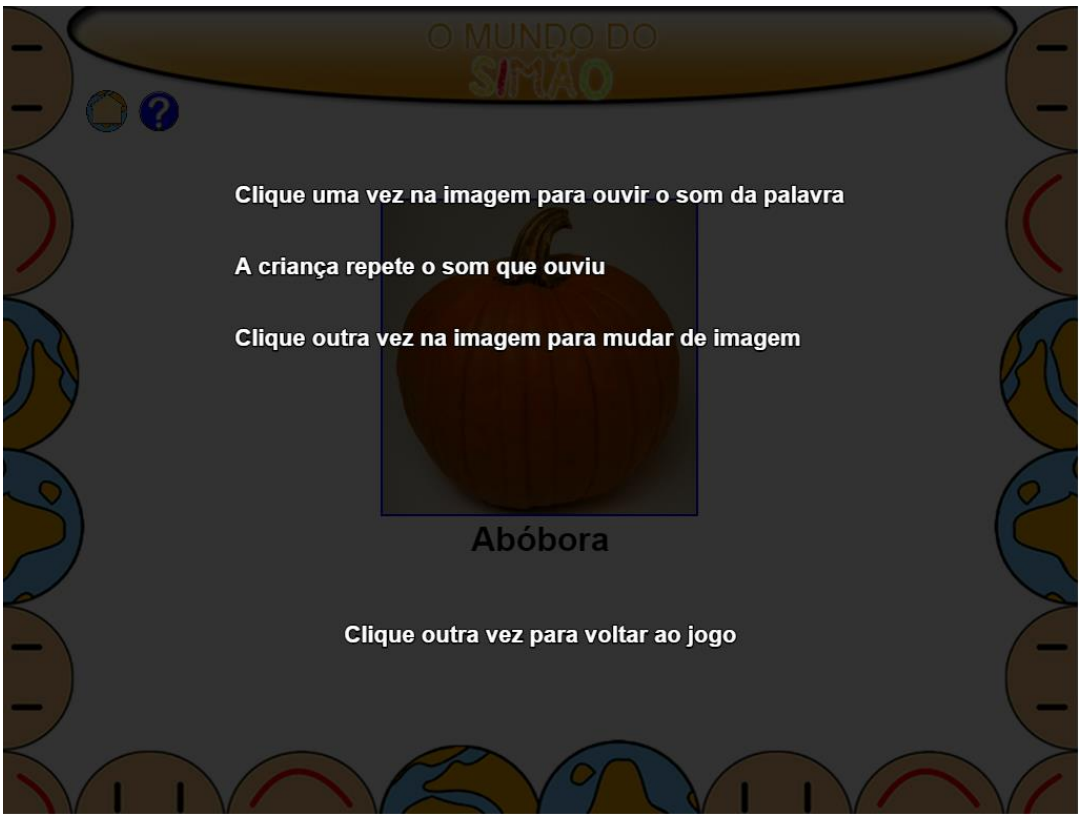
The History of Videogames. (Março, 1982). *Electronic Games*, 20-24.

Valve. (2 de Janeiro de 2016). *Valve*. Obtido de Valve: http://www.valvesoftware.com/games/portal.html

Zynga. (10 de Novembro de 2015). *Zynga | Connecting the World Through Games*. Obtido de Zynga: https://www.zynga.com/

Anexos

Anexo 1:



Anexo 2:

Palavra a Palavra

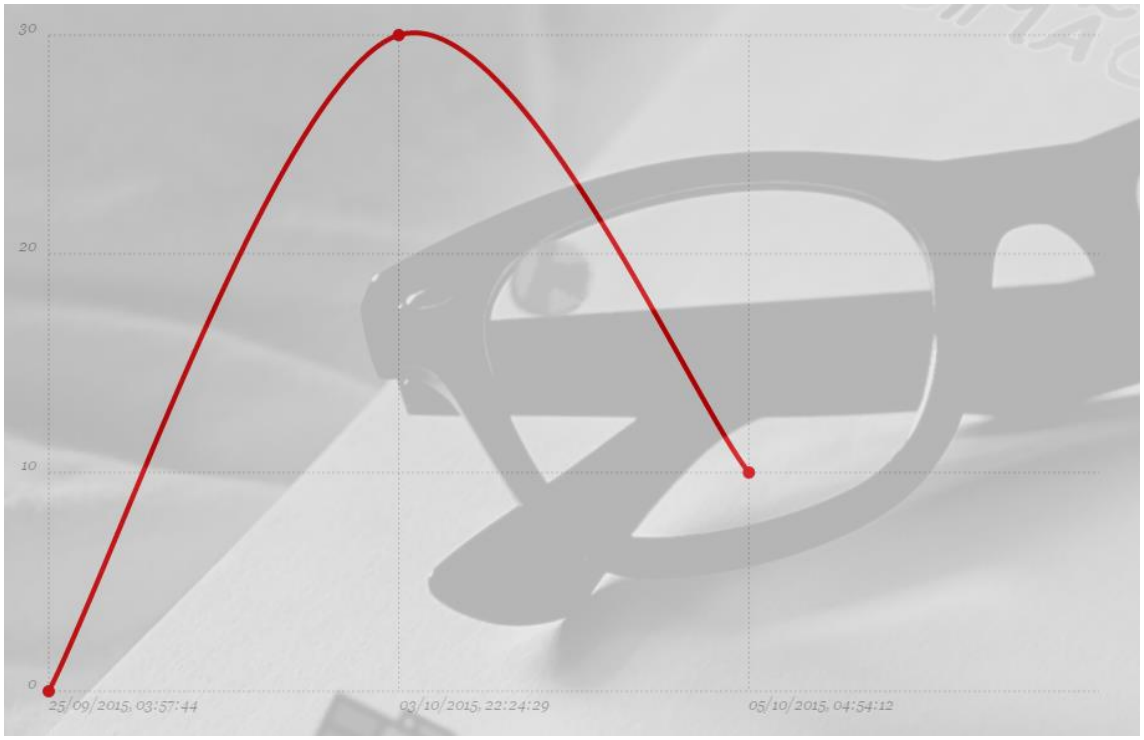
Data De Criacao	Palavra a Palavra
Fri Sep 25 2015 03:57:44 GMT+0100 (GMT Daylight Time)	0
Sat Oct 03 2015 22:24:29 GMT+0100 (GMT Daylight Time)	30
Mon Oct 05 2015 04:54:12 GMT+0100 (GMT Daylight Time)	10

Fraseando

Data De Criacao	Fraseando
Fri Sep 25 2015 03:57:44 GMT+0100 (GMT Daylight Time)	0

Sons Iniciais

Data De Criacao	Sons Iniciais
Fri Sep 25 2015 03:57:44 GMT+0100 (GMT Daylight Time)	0

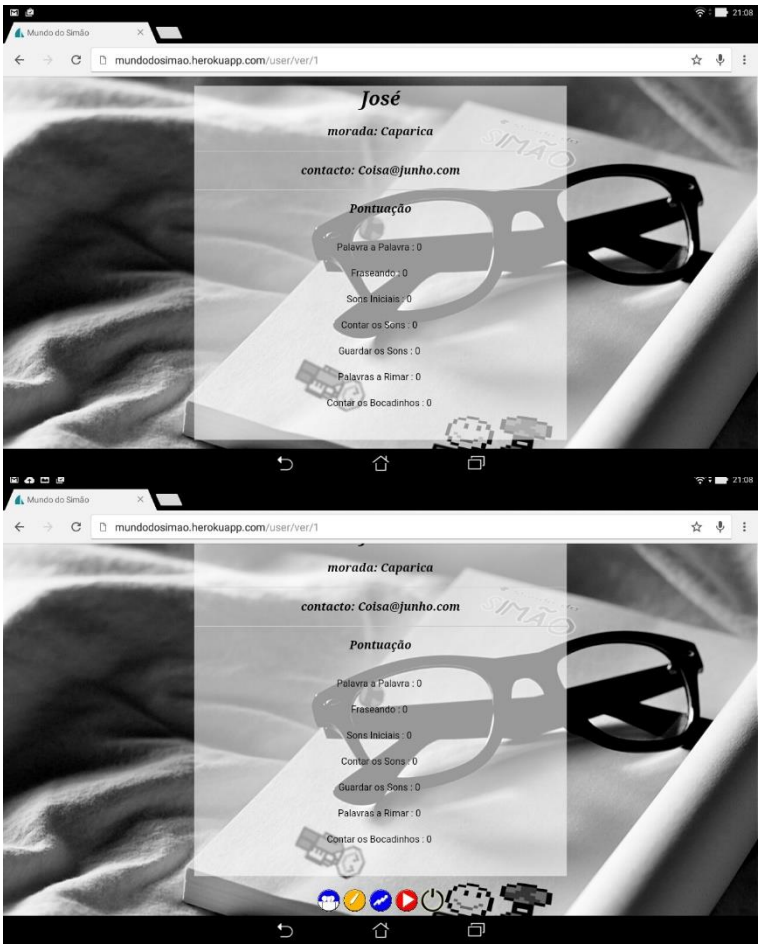


Anexo 3:

Utilizadores

ID	Nome	Morada	Email			
1	Luis	Rua do Sol	luis@luisinho.com	Ver	Editar	Delete
2	Carlos	Avenida Antiga	c.arlos@vida.pt	Ver	Editar	Delete
3	Paulo	Avenida da Liberdade	p.liberdade@yahoos.com	Ver	Editar	Delete
4	Rodrigo	Rua do Campo	rodrig@mail.pt	Ver	Editar	Delete
5	Maria	Avenida de Marte	maria@marciana.pt	Ver	Editar	Delete
6	Hélio	Rotunda da Calçada	helio@rotundacalc.com	Ver	Editar	Delete
7	Vasco	Avenida da Antena 10	vasco@naantena10.pt	Ver	Editar	Delete
8	Bob	Rua Inglesa	bob@inglesadarua.pt	Ver	Editar	Delete
9	Pedro	Calçada Generosa	calcada@pedrogen.com	Ver	Editar	Delete
10	Sara	Rua da Poesia	poeta@sara.pt	Ver	Editar	Delete
11	Nádia	Avenida Holandesa	holanda@nádia.pl	Ver	Editar	Delete
12	António	Rua Sicale	sicale@cafedoantonio.com	Ver	Editar	Delete
13	Marilia	Avenida Agricola	campo@mariliafarm.com	Ver	Editar	Delete
14	João	Rua Delicada	delicadeza@eojoao.com	Ver	Editar	Delete
15	Rúben	Avenida da Fé	rubendefe@igreja.com	Ver	Editar	Delete

Anexo 4:



Anexo 5: